How This Help File is Organized

VBA and CorelDRAW 10.0

CorelDRAW 10.0 VBA Help outlines the key components of the Visual Basic for Applications (VBA) programming environment as it relates to the CorelDRAW 10.0 application.

VBA is a subset of the Microsoft Visual Basic (VB) object-oriented programming environment. VBA uses the Visual Basic Editor interactive development environment and the VB programming language to enhance applications by manipulating the application's objects, exposed by its object model. VBA can access other applications from within CorelDRAW 10.0 by referencing that application's object model components.

CorelDRAW 10.0 VBA Help assumes that you have a basic knowledge of VBA and the VB programming environment. For more detailed instruction relating to these topics, please consult the "Microsoft Visual Basic for Applications Help" from the Help menu in the Visual Basic Editor.

Object Model Reference

CorelDRAW 10.0 VBA Help divides its object model references into several categories:

- classes
- enums
- properties
- methods
- events
- constants

Topics within each category are listed in alphabetical order for easy navigation and contain links to other categories as they relate to the current topic.

Sample Code

CorelDRAW 10.0 VBA Help provides sample code for all properties, methods and events in the CorelDRAW 10.0 object model. Code samples are located in the main topic window for each property, method and event.

The **CorelDRAW** application object is not referenced in the sample code. The application object does not need to be included when referencing objects within the application. If you are referencing the CorelDRAW 10.0 application with VBA from another application, the reference must include the CorelDRAW application object.

Introduction to Visual Basic for Applications

What is Visual Basic for Applications?

Visual Basic for Applications (VBA) is a subset of the Microsoft Visual Basic (VB) object-oriented programming environment. VBA uses the Visual Basic Editor interactive development environment and the VB programming language to enhance applications by manipulating the application's objects, exposed by its object model. VBA can access other applications from within Corel PHOTO-PAINT 10.0 by referencing that application's object model components.

VBA provides you with a set of tools that you can use to customize the graphical user interface of Corel applications, such as CorelDRAW and Corel PHOTO-PAINT. These tools allow you to process information and present data in an efficient and effect forum. Even though VBA uses the Visual Basic programming language, it is considered "for applications" because it is most often integrated into another application in order to customize the functionality of that application.

VBA allows you to record and run macros that automate a series of tasks within the Corel application.

What is the difference between Visual Basic, Visual Basic for Applications and VBScript?

The Microsoft Visual Basic programming system is an advanced set of programming tools that provides advanced functionality and components for the Microsoft Windows operating system and other windows-based programs. For example, with Visual Basic you can create application extensions (dll's) and stand-alone executable programs (exe's). You cannot create either of these components with VBA or VBScript.

VBA is also referred to as Visual Basic, Applications Edition. VBA is a subset of the Visual Basic programming language. It uses the programming structure of Visual Basic to manipulate objects of an object model, left exposed by an application like CorelDRAW or Corel PHOTO-PAINT. The manipulation of these objects results in small packets of code procedures within the application. These code procedures and resulting projects are called add ins.

VBScript is also referred to as Microsoft Visual Basic, Scripting Edition. VBScript is also a subset of the Visual Basic programming language. It is a web-based HTML document scripting language.

The Visual Basic Editor

Each document or image that you create with VBA has a corresponding Visual Basic for Applications project. For example, when you open a document in CorelDRAW, an accompanying Visual Basic project is opened up in the Visual Basic Editor. In order to customize your document with VBA coding procedures, you must open the project file in the Visual Basic Editor. To display the Editor, go to Tools|Visual Basic|Visual Basic Editor on the main menu in the application.

For more detailed information on constructing code procedures in the Visual Basic Editor, view the Help Topics located within the Visual Basic Editor.

VBA Programming Components

What is an Object Model?

The VBA programming language automates tasks within an application, by manipulating the components of the application's object model. An object model represents the hierarchy of objects within an application and their relationship to each other within the paradigm.

In CorelDRAW and Corel PHOTO-PAINT, the **Application** object represents the beginning of the object hierarchy. Starting with the Application object, you drill down and navigate through the object model until you find the desired object. To reference an object with Visual Basic code, you separate each level of the object hierarchy with the dot operator (.).

For example, "CorelDRAW. ActiveDocument" references the **Document** object, but separates the document class from the application class with the use of the dot operator. It is important to note that you do not need to include the application object when referencing objects in the object model of CorelDRAW and Corel PHOTO-PAINT.

What is the difference between a Class and an Object?

A class is a definition of an object, not a representation of the actual object. A class outlines the properties, methods, and events that apply to a type of object in CorelDRAW and Corel PHOTO-PAINT. It acts as a template for all objects of that type class. An object is an instance of a class.

For example, "Document" represents the **Document** class in Corel applications. However, "ActiveDocument" represents an object within that class because it makes specific reference to one object.

What is a Collection?

A collection is a group of objects that are similar in type. As objects, they share the same properties, methods, and events. They are uniquely identified within the collection by their index number or their name. Collection objects act in the same manner and are always plural.

For example, "Documents" represent the **Documents** collection class in Corel applications. However, "Documents.Item (1)" references the first document object in the collection.

What is a Property?

Each object within an object model is defined by a property, method, event, or a combination of each. A property is an adjective, and acts as an object attribute. It represents a characteristic quality of the object. Properties can be returned, set, or read-only. The provide information about the object.

For example, "ActiveDocument.Name" represents the **Name** property of a Document object.

What is a Method?

Each object within an object model is defined by a property, method, event, or a combination of each. A method is a verb, and is seen as the action of an object. It represents an act that can be performed by an object.

For example, "ActiveDocument.Close" represents the **Close** method of a Document object. It performs the act of closing the document in the application.

What is an Event?

Each object within an object model is defined by a property, method, event, or a combination of each. An event is a noun, and acts as something that takes place in an object. You write code for an object to respond to the act. Events are triggered by an action, such as a click, key press or system timer.

For example, "ActiveDocument.AfterSave" event triggers an action in the **Document** object after it has been saved.

What is the difference between Enumerations and Constants?

Enumerations and Constants both represent fixed values in VBA programming structures. Unlike a variable, which temporarily stores a changing data value in a code procedure or function, enumerations and constants both represent fixed values in code procedures and functions - their values do not change.

An enumeration groups similar constants together. For Example, "AddinFilter" is an enumeration, yet it contains several constants, including "AddinFilterNone" and AddinFilterNew". A constant is an instance of an enumeration.

GlobalDocument events

GlobalDocument

AfterPrint
AfterSave
AfterSaveAs
BeforeClose
BeforePrint
BeforeSave
BeforeSaveAs
CorelScriptStartPlaying
CorelScriptStartRecording
CorelScriptStopPlaying
CorelScriptStopRecording
LayerCreated
LayerDeleted
LayerSelected
NewDocument
OpenDocument
PageCreated
PageDeleted
PageSelected
Quit
SelectionChanged
ShapeCreated
ShapeDeleted
ShapeMoved
Start

GlobalDocument

Class **GlobalDocument**

Events Referenced By

{button ,AL(`CLS_GlobalDocument')} **Related Topics**

GlobalDocument.AfterPrint

Event **AfterPrint**()

Event triggered after the Print dialog closes

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_AfterPrint')} **Related Topics**

GlobalDocument.AfterSave

Event **AfterSave**(ByVal **FileName** As String)

Event triggered after the save is completed

Member of **GlobalDocument**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_AfterSave')} **Related Topics**

GlobalDocument.AfterSaveAs

Event **AfterSaveAs**(ByVal **FileName** As String)

Event triggered after the save is completed

Member of **GlobalDocument**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_AfterSaveAs')} **Related Topics**

GlobalDocument.BeforeClose

Event **BeforeClose**()

Event triggered before the document closes

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_BeforeClose')} **Related Topics**

GlobalDocument.BeforePrint

Event **BeforePrint**()

Event triggered before the Print dialog appears

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_BeforePrint')} **Related Topics**

GlobalDocument.BeforeSave

Event **BeforeSave**()

Event triggered before the save dialog appears

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_BeforeSave')} **Related Topics**

GlobalDocument.BeforeSaveAs

Event **BeforeSaveAs**()

Event triggered before the save dialog appears

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_BeforeSaveAs')} **Related Topics**

GlobalDocument.CorelScriptStartPlaying

Event **CorelScriptStartPlaying**()

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_CorelScriptStartPlaying')} **Related Topics**

GlobalDocument.CorelScriptStartRecording

Event **CorelScriptStartRecording**()

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_CorelScriptStartRecording')} **Related Topics**

GlobalDocument.CorelScriptStopPlaying

Event **CorelScriptStopPlaying**()

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_CorelScriptStopPlaying')} **Related Topics**

GlobalDocument.CorelScriptStopRecording

Event **CorelScriptStopRecording**(ByVal **FileName** As String)

Member of **GlobalDocument**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_CorelScriptStopRecording')} **Related Topics**

GlobalDocument.LayerCreated

Event **LayerCreated**(ByRef **Layer** As **Layer**)

Event triggered after layer creation

Member of **GlobalDocument**

Parameters	Description
Layer	Description of Layer goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_LayerCreated')} **Related Topics**

GlobalDocument.LayerDeleted

Event **LayerDeleted**(ByRef **Layer** As **Layer**)

Event triggered after layer deletion

Member of **GlobalDocument**

Parameters	Description
Layer	Description of Layer goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_LayerDeleted')} **Related Topics**

GlobalDocument.LayerSelected

Event **LayerSelected**(ByRef **Layer** As **Layer**)

Event triggered after layer selection

Member of **GlobalDocument**

Parameters	Description
Layer	Description of Layer goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_LayerSelected')} **Related Topics**

GlobalDocument.NewDocument

Event **NewDocument**(ByVal **Document** As **Document**)

Member of **GlobalDocument**

Parameters	Description
Document	Description of Document goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_NewDocument')} **Related Topics**

GlobalDocument.OpenDocument

Event **OpenDocument**(ByVal **Document** As **Document**)

Member of **GlobalDocument**

Parameters	Description
Document	Description of Document goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_OpenDocument')} **Related Topics**

GlobalDocument.PageCreated

Event **PageCreated**(ByRef **Page** As **Page**)

Event triggered after page creation

Member of **GlobalDocument**

Parameters	Description
Page	Description of Page goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_PageCreated')} **Related Topics**

GlobalDocument.PageDeleted

Event **PageDeleted**(ByRef **Page** As **Page**)

Event triggered after page deletion

Member of **GlobalDocument**

Parameters	Description
Page	Description of Page goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_PageDeleted')} **Related Topics**

GlobalDocument.PageSelected

Event **PageSelected**(ByRef **Page** As **Page**)

Event triggered after page selection

Member of **GlobalDocument**

Parameters	Description
Page	Description of Page goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_PageSelected')} **Related Topics**

GlobalDocument.Quit

Event **Quit**()

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_Quit')} **Related Topics**

GlobalDocument.SelectionChanged

Event **SelectionChanged**(ByRef **Shape** As **Shape**)

Event triggered after selection changed

Member of **GlobalDocument**

Parameters	Description
Shape	Description of Shape goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_SelectionChanged')} **Related Topics**

GlobalDocument.ShapeCreated

Event **ShapeCreated**(ByRef **Shape** As **Shape**)

Event triggered after shape creation

Member of **GlobalDocument**

Parameters	Description
Shape	Description of Shape goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_ShapeCreated')} **Related Topics**

GlobalDocument.ShapeDeleted

Event **ShapeDeleted**(ByRef **Shape** As **Shape**)

Event triggered after shape deletion

Member of **GlobalDocument**

Parameters	Description
Shape	Description of Shape goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_ShapeDeleted')} **Related Topics**

GlobalDocument.ShapeMoved

Event **ShapeMoved**(ByRef **Shape** As **Shape**)

Event triggered after shape creation

Member of **GlobalDocument**

Parameters	Description
Shape	Description of Shape goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_ShapeMoved')} **Related Topics**

GlobalDocument.Start

Event **Start**()

Member of **GlobalDocument**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GlobalDocument;FNC_Start')} **Related Topics**

StructFountainFillProperties properties

StructFountainFillProperties Legend

- Angle
- BlendType
- CenterX
- CenterY
- ColorCount
- EdgePad
- FromColor
- MidPoint
- Steps
- ToColor
- Type

StructFountainFillProperties

Class **StructFountainFillProperties**

Properties Referenced By

{button ,AL(`CLS_StructFountainFillProperties')} **Related Topics**

StructFountainFillProperties.Angle

Property **Angle** As Double

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_Angle')} **Related Topics**

StructFountainFillProperties.BlendType

Property **BlendType** As **cdrFountainFillBlendType**

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_BlendType')} **Related Topics**

StructFountainFillProperties.CenterX

Property **CenterX** As Long

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_CenterX')} **Related Topics**

StructFountainFillProperties.CenterY

Property **CenterY** As Long

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_CenterY')} **Related Topics**

StructFountainFillProperties.ColorCount

Property **ColorCount** As Long

Member of **StructFountainFillProperties**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_ColorCount')} **Related Topics**

StructFountainFillProperties.EdgePad

Property **EdgePad** As Long

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_EdgePad')} **Related Topics**

StructFountainFillProperties.FromColor

Property **FromColor** As **Color**

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_FromColor')} **Related Topics**

StructFountainFillProperties.MidPoint

Property **MidPoint** As Long

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_MidPoint')} **Related Topics**

StructFountainFillProperties.Steps

Property **Steps** As Long

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_Steps')} **Related Topics**

StructFountainFillProperties.ToColor

Property **ToColor** As **Color**

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_ToColor')} **Related Topics**

StructFountainFillProperties.Type

Property **Type** As **cdrFountainFillType**

Member of **StructFountainFillProperties**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructFountainFillProperties;FNC_Type')} **Related Topics**

CommandBarMode properties

CommandBarMode Legend

- Controls
- Name
- NameLocal

CommandBarMode

Class **CommandBarMode**

Properties Referenced By

The **CommandBarMode** class defines the characteristics of command bar mode and describes the look and behavior of the object through its properties and methods. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

{button ,AL(`CLS_CommandBarMode')} **Related Topics**

CommandBarMode.Controls

Property **Controls** As **CommandBarControls**

Description

The **Controls** property returns a value associated with the **CommandBarControls** collection in CorelDRAW. This property returns a value associated with the controls that are available when a command bar in CorelDRAW is in a specific mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

{button ,AL(`CLS_CommandBarMode;FNC_Controls')} **Related Topics**

CommandBarMode.Name

Property **Name** As String

Description

The **Name** property returns a string value associated with name of a command bar mode in CorelDRAW. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

{button ,AL(`CLS_CommandBarMode;FNC_Name')} **Related Topics**

CommandBarMode.NameLocal

Property **NameLocal** As String

Description

The **Name** property returns a string value associated with localized name of a command bar mode in CorelDRAW. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

{button ,AL(`CLS_CommandBarMode;FNC_NameLocal')} **Related Topics**

CommandBarModes properties

CommandBarModes Legend

- Count

- Item

CommandBarModes

Class **CommandBarModes**

Properties Referenced By

The **CommandBarModes** class defines the characteristics of command bar mode objects and describes the look and behavior of the object through its properties and methods. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

{button ,AL(`CLS_CommandBarModes')} **Related Topics**

CommandBarModes.Count

Property **Count** As Long

Description

The **Count** property counts the number of command bar modes in a command bar of CorelDRAW. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

{button ,AL(`CLS_CommandBarModes;FNC_Count')} **Related Topics**

CommandBarModes.Item

Property **Item**(ByVal **IndexOrName** As Variant) As **CommandBarMode**

Description

The **Index** property returns a value associated with a command bar mode in a command bar of CorelDRAW. A mode is the state or setting of the command bar in CorelDRAW. For example, when a word processor is in insert mode, characters that you type are inserted at the cursor position. In overstrike mode, characters typed replace existing characters. The term mode implies a choice -- that you can change the setting and put the system in a different mode.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **CommandBarModes** collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each command bar mode.

{button ,AL(`CLS_CommandBarModes;FNC_Item')} **Related Topics**

DataField properties

DataField Legend

- Application
  - DocDefault
  - DrawDefault
  - FieldWidth
  - Format
- FormatType
- Index
  - Name
- Parent
  - SummarizeGroup

DataField methods

DataField Legend

- Delete
- Reorder

DataField

Class **DataField**

Properties Methods Referenced By

The **DataField** class defines the characteristics of data field objects and describes the look and behavior of the object through its properties and methods.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

You can change the name of any field to better suit your object data summary. You can also change the location of fields to make them appear in a logical order on the data summary.

You can delete any data field except for CDRStaticID. When you delete a field, you also delete all data entered for that field in the active document.

{button ,AL(`CLS_DataField')} **Related Topics**

DataField.Application

Property **Application** As **Application**

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_DataField;FNC_Application')} **Related Topics**

DataField.DocDefault

Property **DocDefault** As Boolean

Description

The **DocDefault** property returns or sets a True or False value that indicates whether or not a data field is considered a default data field, available in the current document in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

{button ,AL(`CLS_DataField;FNC_DocDefault')} **Related Topics**

DataField.DrawDefault

Property **DrawDefault** As Boolean

Description

The **DrawDefault** property returns or sets a True or False value that indicates whether or not a data field is considered a default data field, available to all documents in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

{button ,AL(`CLS_DataField;FNC_DrawDefault')} **Related Topics**

DataField.FieldWidth

Property **FieldWidth** As Long

Description

The **FieldWidth** property returns or sets the width of a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

{button ,AL(`CLS_DataField;FNC_FieldWidth')} **Related Topics**

DataField.Format

Property **Format** As String

Description

The **Format** property returns or sets the format of a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

{button ,AL(`CLS_DataField;FNC_Format')} **Related Topics**

DataField.FormatType

Property **FormatType** As **cdrDataFormatType**

Description

The **FormatType** property returns the format type of a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

This property is Read-Only and returns a value of cdrDataFormatType.

{button ,AL(`CLS_DataField;FNC_FormatType')} **Related Topics**

DataField.Index

Property **Index** As Long

Description

The **Index** property returns the index number of a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

This property returns a Read-Only value.

{button ,AL(`CLS_DataField;FNC_Index')} **Related Topics**

DataField.Name

Property **Name** As String

Description

The **Name** property returns a string value that uniquely identifies a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

{button ,AL(`CLS_DataField;FNC_Name')} **Related Topics**

DataField.Parent

Property **Parent** As **DataFields**

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_DataField;FNC_Parent')} **Related Topics**

DataField.SummarizeGroup

Property **SummarizeGroup** As Boolean

Description

The **SummarizeGroup** property returns or sets a True or False value that indicates whether or not to summarize a data field group in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

You can display individual group subtotals for fields sharing multiple groups. Use this command when more than one group of objects is displayed in a datasheet. This command applies only to fields with numeric formats.

{button ,AL(`CLS_DataField;FNC_SummarizeGroup')} **Related Topics**

DataField.Delete

Sub **Delete**()

Description

The **Delete** method deletes a data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

{button ,AL(`CLS_DataField;FNC_Delete')} **Related Topics**

DataField.Reorder

Sub **Reorder**(ByVal **NewIndex** As Long)

Description

The **Reorder** method reorders the position of a data field by assigning it a new index number in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

Parameters	Description
NewIndex	**Index** is a preset placeholder for each object in a **DataFields** collection; it uniquely identifies each member of the collection

{button ,AL(`CLS_DataField;FNC_Reorder')} **Related Topics**

DataFields properties

DataFields Legend

- Application
- Count
- Item
- Parent

DataFields methods

DataFields Legend

Add

DataFields

Class **DataFields**

Properties Methods Referenced By

The **DataFields** class defines the characteristics of **DataFields** collection objects and describes the look and behavior of the object through its properties and methods.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

You can change the name of any field to better suit your object data summary. You can also change the location of fields to make them appear in a logical order on the data summary.

You can delete any data field except for CDRStaticID. When you delete a field, you also delete all data entered for that field in the active document.

{button ,AL(`CLS_DataFields')} **Related Topics**

DataFields.Application

Property **Application** As **Application**

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_DataFields;FNC_Application')} **Related Topics**

DataFields.Count

Property **Count** As Long

Description

The **Count** property returns the number of data field objects in the DataFields collection of CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

This property returns a Read-Only value.

{button ,AL(`CLS_DataFields;FNC_Count')} **Related Topics**

DataFields.Item

Property **Item**(ByVal **IndexOrName** As Variant) As **DataField**

Description

The **Item** property returns a value associated with a specific data field in the DataFields collection, by referencing the data field's index number or name in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

This property returns a Read-Only value.

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **DataFields** collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each command bar.

{button ,AL(`CLS_DataFields;FNC_Item')} **Related Topics**

DataFields.Parent

Property **Parent** As **Document**

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_DataFields;FNC_Parent')} **Related Topics**

DataFields.Add

Function **Add**(ByVal **Name** As String, [ByVal Format As String], [ByVal DrawDefault As Boolean = False], [ByVal DocDefault As Boolean = False], [ByVal SummarizeGroup As Boolean = False]) As **DataField**

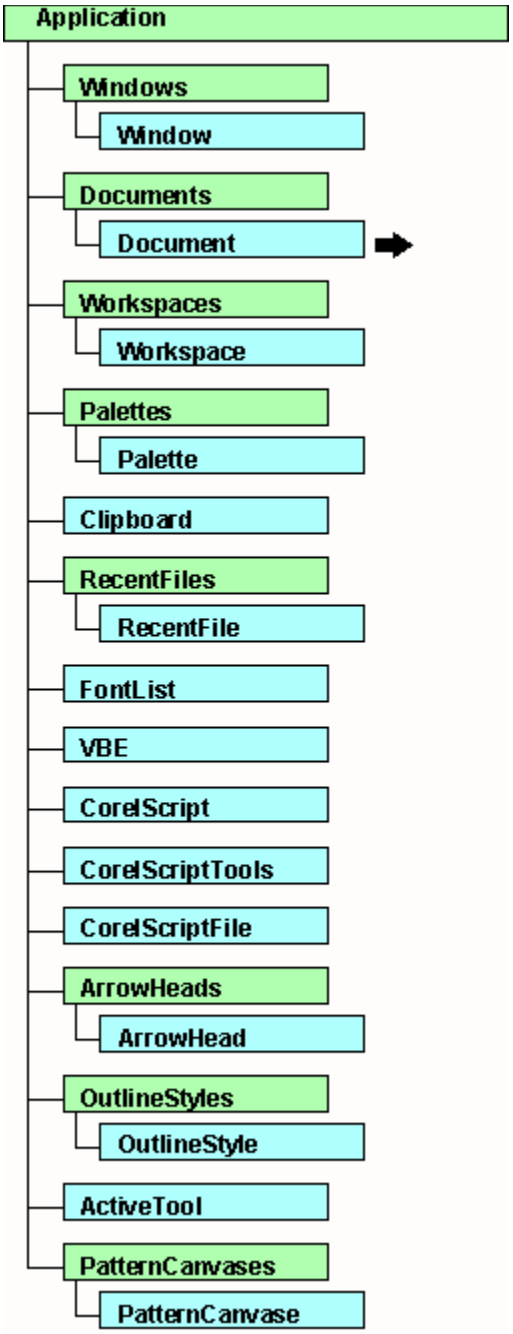
Description

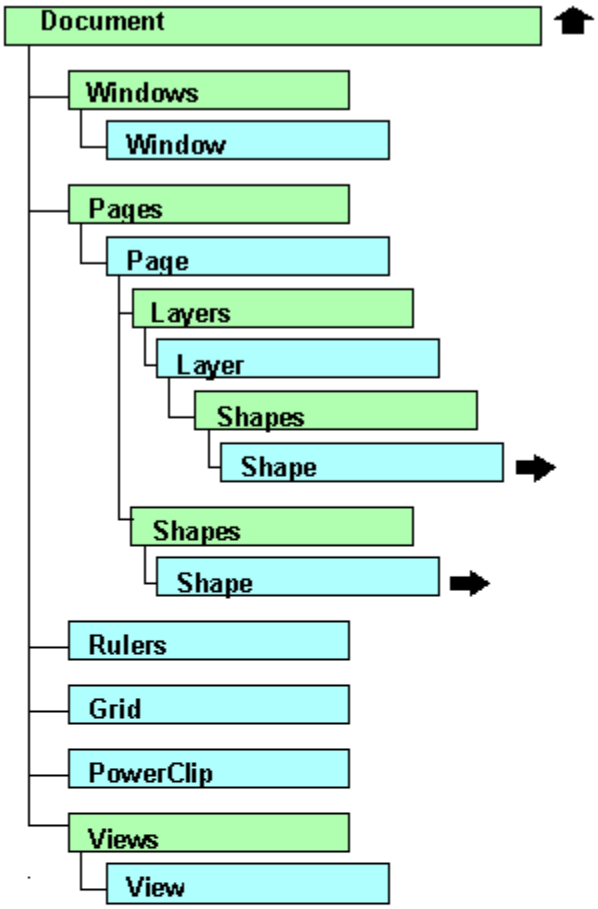
The **Add** method adds a new data field in CorelDRAW. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

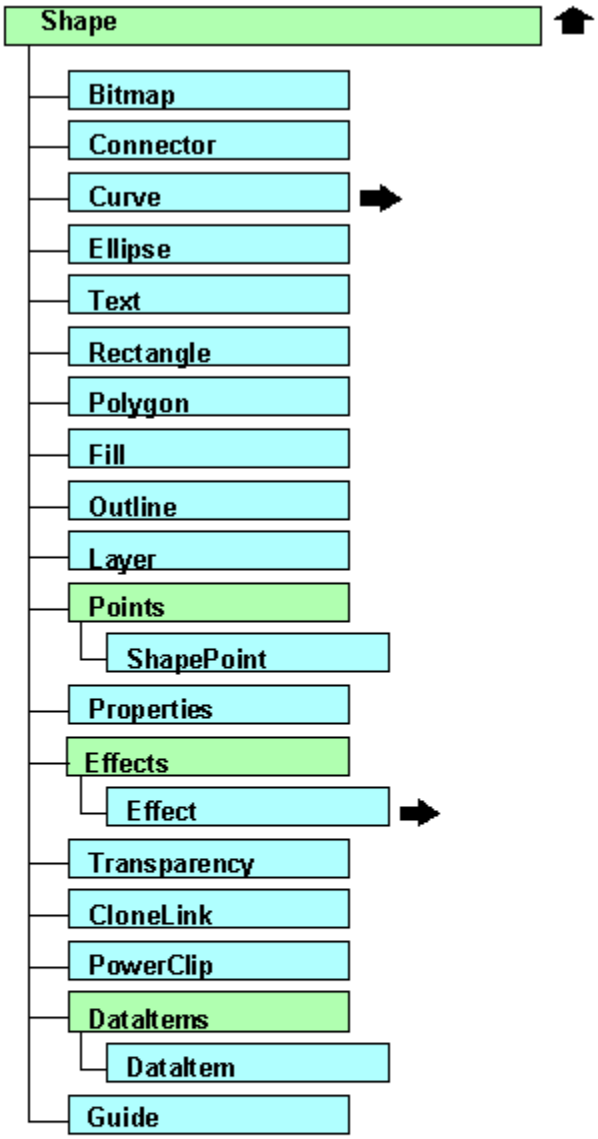
Parameters	Description
Name	Sets the name of the data field. This value is a string value.
Format	Sets the format value of the data field. This value is optional.
DrawDefault	Sets the data field as a default field in CorelDRAW. This value is optional and the default value is False.
DocDefault	Sets the data field as a default field in the document of CorelDRAW. This value is optional and the default value is False.
SummarizeGroup	Sets a True or False value indicating whether or not to summarize the data field. This value is optional and the default value is False.

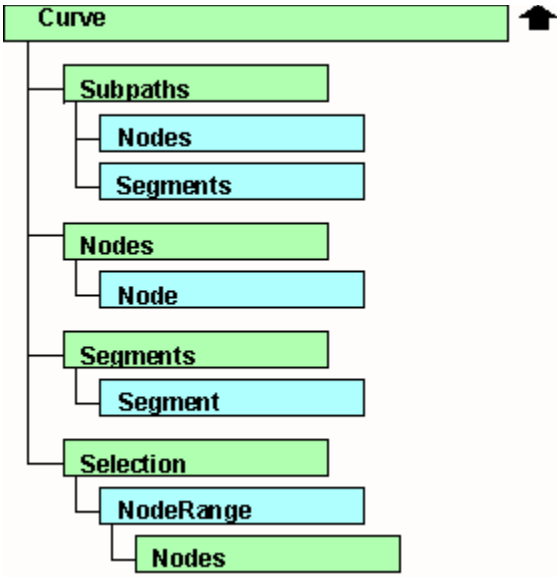
{button ,AL(`CLS_DataFields;FNC_Add')} **Related Topics**

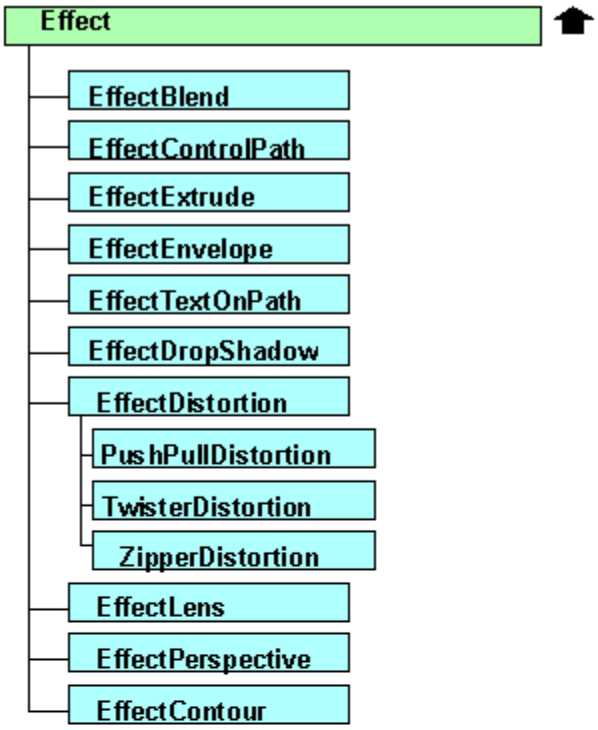
An object that can be displayed as several variations of a rectangle or a circle


Application
Windows
Window
Documents
Document
Workspaces
Workspace
Palettes
Palette
Clipboard
RecentFiles
RecentFile
FontList
VBE
CorelScript
CorelScriptTools
CorelScriptFile
ArrowHeads
ArrowHead
OutlineStyles
OutlineStyle
ActiveTool
PatternCanvases
PatternCanvase


Document
Windows
Window
Pages
Page
Layers
Layer
Shapes
Shape
Shapes
Shape
Rulers
Grid
PowerClip
Views
View


Shape
Bitmap
Connector
Curve
Ellipse
Text
Rectangle
Polygon
Fill
Outline
Layer
Points
ShapePoint
Properties
Effects
Effect
Transparency
CloneLink
PowerClip
DataItems
DataItem
Guide


Curve
Subpaths
Nodes
Segments
Nodes
Node
Segments
Segment
Selection
NodeRange
Nodes


Effect
EffectBlend
EffectControlPath
EffectExtrude
EffectEnvelope
EffectTextOnPath
EffectDropShadow
EffectDistortion
PushPullDistortion
TwisterDistortion
ZipperDistortion
EffectLens
EffectPerspective
EffectContour

Legend

- Default property
- Read-Only property

Legend

- Default method

Rectangle properties

Rectangle Legend

- CornerLowerLeft
- CornerLowerRight
- CornerUpperLeft
- CornerUpperRight
- EqualCorners

- MaxRadius
  - RadiusLowerLeft
  - RadiusLowerRight
  - RadiusUpperLeft
  - RadiusUpperRight

Rectangle methods

Rectangle Legend

- SetRadius
- SetRoundness

Rectangle

Class **Rectangle**

Properties Methods Referenced by

The **Rectangle** class defines the characteristics of rectangle objects and describes the look and behavior of the objects through its properties and methods. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

CorelDRAW provides drawing tools for drawing basic shapes, such as rectangles, ellipses, polygons, stars, grids, and spirals. To draw a shape with one of these tools, drag diagonally in any direction until the shape is the size you want. For each tool, the Status Bar displays the dimensions of the shape as you draw it.

If you hold down SHIFT, CorelDRAW creates the rectangle or square from the center outward as you drag. Double-clicking the Rectangle tool creates a rectangle covering the entire drawing page. This is useful if you want to create a background for the drawing.

{button ,AL(`CLS_Rectangle')} **Related Topics**

Rectangle.CornerUpperLeft

Property **CornerUpperLeft** AS Long

Rectangle

Description

The **CornerUpperLeft** property returns or sets a numerical value that indicates the roundness of a rectangle's upper left corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_CornerUpperLeft')} **Related Topics**

Rectangle.CornerUpperRight

Property **CornerUpperRight** AS Long

Rectangle

Description

The **CornerUpperRight** property returns or sets a numerical value that indicates the roundness of a rectangle's upper right corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_CornerUpperRight')} **Related Topics**

Rectangle.CornerLowerLeft

Property **CornerLowerLeft** AS Long

Rectangle

Description

The **CornerLowerLeft** property returns or sets a numerical value that indicates the roundness of a rectangle's lower left corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_CornerLowerLeft')} **Related Topics**

Rectangle.CornerLowerRight

Property **CornerLowerRight** AS Long

Rectangle

Description

The **CornerLowerRight** property returns or sets a numerical value that indicates the roundness of a rectangle's lower right corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_CornerLowerRight')} **Related Topics**

Rectangle.EqualCorners

Property **EqualCorners** AS Boolean

Rectangle

Description

The **EqualCorners** property returns a True or False value indicating if a rectangle's corners are equal in size and shape. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

You can round the corners of a rectangle with precision by entering values in the Rectangle Corner Roundness boxes on the Property Bar. To round only one corner, unlock the Round Corners Together button. If the **EqualCorners** property is True, the Round Corners Together button is enabled and all rectangle corners will be the same size.

The **EqualCorners** property returns a Read-Only value.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_EqualCorners')} **Related Topics**

Rectangle.MaxRadius

Property **MaxRadius** AS Double

Rectangle

Description

The **MaxRadius** property returns the maximum radius value for a rectangle's rounded corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc. A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

The **MaxRadius** returns a Read-Only value.

Example

The following code example displays the maximum radius value in the active rectangle shape:

```
Sub Max()
MsgBox ActiveShape.Rectangle.MaxRadius
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_MaxRadius')} **Related Topics**

Rectangle.SetRoundness

Sub **SetRoundness**(ByVal **Roundness** AS Long)

Rectangle

Description

The **SetRoundness** method sets the level of roundness the corners of a rectangle in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

You can round the corners of a rectangle with precision by entering values in the Rectangle Corner Roundness boxes on the Property Bar. To round only one corner, unlock the Round Corners Together button. Applying the roundness of a rectangle's corners with the **SetRoundness** method sets the same roundness scale for all corners in the rectangle.

In order to set the roundness of a rectangle's corners with the **SetRoundness** method, you must pass the **Roundness** value as a parameter:

Parameters	Description
Roundness	The **Roundness** parameter is a numerical value that sets the degree of roundness for all corners in a rectangle shape in CorelDRAW. The values can range from 0 to 100.

Example

The following code example sets the roundness value for all corners in the active rectangle shape of CorelDRAW:

```
Sub SetRound()
With ActiveShape.Rectangle
    .SetRoundness 80
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_SetRoundness')} **Related Topics**

Rectangle.SetRadius

Sub **SetRadius**(ByVal **Radius** AS Double)

Rectangle

Description

The **SetRadius** method sets the radius value in the corners of a rectangle in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar. The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

When you set the radius of a rectangle's corners with the **SetRadius** method, you are setting the same radius value for all corners in the rectangle.

In order to set the radius of a rectangle's corners with the **SetRadius** method, you must pass the **Radius** value as a parameter:

Parameters	Description
Radius	The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

Example

The following code example sets the radius value in the active rectangle shape of CorelDRAW:

```
Sub SetRound()
With ActiveShape.Rectangle
    .SetRadius .05
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_SetRadius')} **Related Topics**

Rectangle.RadiusUpperLeft

Property **RadiusUpperLeft** AS Double

Rectangle

Description

The **RadiusUpperLeft** property returns or sets a numerical value that indicates the radius of a rectangle's upper left corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

Example

The following code example sets the radius value for each corner in the active rectangle shape of CorelDRAW:

```
Sub SetRad()
With ActiveShape.Rectangle
    .RadiusLowerLeft = .02
    .RadiusLowerRight = .02
    .RadiusUpperRight = .02
    .RadiusUpperLeft = .02
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_RadiusUpperLeft')} **Related Topics**

Rectangle.RadiusUpperRight

Property **RadiusUpperRight** AS Double

Rectangle

Description

The **RadiusUpperRight** property returns or sets a numerical value that indicates the radius of a rectangle's upper right corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

Example

The following code example sets the radius value for each corner in the active rectangle shape of CorelDRAW:

```
Sub SetRad()
With ActiveShape.Rectangle
    .RadiusLowerLeft = .02
    .RadiusLowerRight = .02
    .RadiusUpperRight = .02
    .RadiusUpperLeft = .02
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_RadiusUpperRight')} **Related Topics**

Rectangle.RadiusLowerLeft

Property **RadiusLowerLeft** AS Double

Rectangle

Description

The **RadiusLowerLeft** property returns or sets a numerical value that indicates the radius of a rectangle's lower left corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

Example

The following code example sets the radius value for each corner in the active rectangle shape of CorelDRAW:

```
Sub SetRad()
With ActiveShape.Rectangle
    .RadiusLowerLeft = .02
    .RadiusLowerRight = .02
    .RadiusUpperRight = .02
    .RadiusUpperLeft = .02
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_RadiusLowerLeft')} **Related Topics**

Rectangle.RadiusLowerRight

Property **RadiusLowerRight** AS Double

Rectangle

Description

The **RadiusLowerRight** property returns or sets a numerical value that indicates the radius of a rectangle's lower right corner in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A rectangle has a node at each corner. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. You can control the size of this arc by moving any of the corner nodes. The amount of rounding is displayed on the Property Bar.

The radius is the measurement from the middle of the rectangle's rounded corner to its circumference of the corner's rounded arc.

Example

The following code example sets the radius value for each corner in the active rectangle shape of CorelDRAW:

```
Sub SetRad()
With ActiveShape.Rectangle
    .RadiusLowerLeft = .02
    .RadiusLowerRight = .02
    .RadiusUpperRight = .02
    .RadiusUpperLeft = .02
End With
End Sub
```

{button ,AL(`CLS_Rectangle;FNC_RadiusLowerRight')} **Related Topics**

Ellipse properties

Ellipse Legend

CenterXIDH_Ellipse_CenterX>main
CenterYIDH_Ellipse_CenterY>main
ClockwiseIDH_Ellipse_Clockwise>main
EndAngleIDH_Ellipse_EndAngle>main
HRadiusIDH_Ellipse_HRadius>main
StartAngleIDH_Ellipse_StartAngle>main
TypeIDH_Ellipse_Type>main
VRadiusIDH_Ellipse_VRadius>main

Ellipse methods

Ellipse Legend

- GetCenterPosition
- GetRadius
- SetCenterPosition
- SetRadius

Ellipse.GetCenterPosition

Sub **GetCenterPosition**(ByRef **PositionX** As Double, ByRef **PositionY** As Double)

Description

The **GetCenterPosition** method gets the center of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Parameters	Description
PositionX	Gets the x coordinate for the center or an ellipse, measured in document units.
PositionY	Gets the y coordinate for the center or an ellipse, measured in document units.

{button ,AL(`CLS_Ellipse;FNC_GetCenterPosition')} **Related Topics**

Ellipse.GetRadius

Sub **GetRadius**(ByRef **HRadius** As Double, ByRef **VRadius** As Double)

Description

The **GetRadius** method gets the radius value of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Radius is the measurement from the center of the ellipse to its circumference. An ellipse has a horizontal and vertical radius value.

Parameters	Description
HRadius	Gets the horizontal radius of an ellipse, measured in document units.
VRadius	Gets the vertical radius of an ellipse, measured in document units.

{button ,AL(`CLS_Ellipse;FNC_GetRadius')} **Related Topics**

Ellipse.SetCenterPosition

Sub **SetCenterPosition**(ByVal **PositionX** As Double, ByVal **PositionY** As Double)

Description

The **SetCenterPosition** method sets the center of an ellipse in CorelDRAW.An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Parameters	Description
PositionX	Sets the x coordinate for the center or an ellipse, measured in document units.
PositionY	Sets the y coordinate for the center or an ellipse, measured in document units.

{button ,AL(`CLS_Ellipse;FNC_SetCenterPosition')} **Related Topics**

Ellipse.SetRadius

Sub **SetRadius**(ByVal **HRadius** As Double, [ByVal VRadius As Double = 0])

Description

The **SetRadius** method sets the radius value of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Radius is the measurement from the center of the ellipse to its circumference. An ellipse has a horizontal and vertical radius value.

Parameters	Description
HRadius	Sets the horizontal radius of an ellipse, measured in document units.
VRadius	Sets the vertical radius of an ellipse, measured in document units. This value is optional and the default value is 0.

{button ,AL(`CLS_Ellipse;FNC_SetRadius')} **Related Topics**

Ellipse

Class **Ellipse**

Properties Referenced by

The **Ellipse** class defines the characteristics of ellipse objects and describes the look and behavior of the objects through its properties and methods. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

A curve object is an object that can be any shape. Curve objects have nodes (the points on a path that determine its shape) and control points (points that extend from nodes to further define a path's shape) that you manipulate to change the object's shape. Curve objects can be drawn with the Freehand tool, Bezier tool, Spiral tool, and Natural Pen tool. You can also convert text and objects drawn with the Rectangle tool, Ellipse tool, and Polygon tool into curve objects by using the Convert To Curves command in the Arrange menu.

You can use the **Shape** tool to turn an ellipse or circle into an arc or a pie shape. A simple ellipse has one node, but when you create an arc or pie shape, CorelDRAW splits this node in two.

You control the appearance of the arc or the pie shape by moving these two new nodes. You can also change the direction in which CorelDRAW draws arcs and pie shapes.

{button ,AL(`CLS_Ellipse')} **Related Topics**

cdrEllipse=0

cdrArc=1

cdrPie=2

Ellipse.CenterX

Property **CenterX** As Double

Description

The **CenterX** property returns or sets the value of the horizontal, or x, coordinate for the center of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

{button ,AL(`CLS_Ellipse;FNC_CenterX')} **Related Topics**

Ellipse.CenterY

Property **CenterY** As Double

Description

The **CenterY** property returns or sets the value of the vertical, or y, coordinate for the center of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

{button ,AL(`CLS_Ellipse;FNC_CenterY')} **Related Topics**

Ellipse.HRadius

Property **HRadius** As Double

Description

The **HRadius** property returns or sets the horizontal radius value of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Radius is the measurement from the center of the ellipse to its circumference. An ellipse has a horizontal and vertical radius value.

{button ,AL(`CLS_Ellipse;FNC_HRadius')} **Related Topics**

Ellipse.VRadius

Property **VRadius** As Double

Description

The **VRadius** property returns or sets the vertical radius value of an ellipse in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Radius is the measurement from the center of the ellipse to its circumference. An ellipse has a horizontal and vertical radius value.

{button ,AL(`CLS_Ellipse;FNC_VRadius')} **Related Topics**

Ellipse.Type

Property **Type** AS **cdrEllipseType**

Ellipse

Description

The **Type** property returns or sets a value indicating the ellipse type in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

You can use the **Shape** tool to turn an ellipse or circle into an arc or a pie shape. A simple ellipse has one node, but when you create an arc or pie shape, CorelDRAW splits this node in two.

You control the appearance of the arc or the pie shape by moving these two new nodes. You can also change the direction in which CorelDRAW draws arcs and pie shapes.

The **Type** property returns a value of cdrEllipseType.

Example

The following code example determines the size, shape, and appearance of an ellipse in CorelDRAW To determine the size of the ellipse, the Start Angle of the ellipse is set to 10 and the End Angle is set to 15. The new ellipse is created in a clockwise direction by setting the **Clockwise** property to True, and the ellipse type is set to a pie-type ellipse:

```
Sub EllipseType()
With ActiveShape.Ellipse
    .StartAngle = 10
    .EndAngle = 15
    .Type = cdrPie
    .Clockwise = True
End With
End Sub
```

{button ,AL(`CLS_Ellipse;FNC_Type')} **Related Topics**

Ellipse.StartAngle

Property **StartAngle** AS Double

Ellipse

Description

The **StartAngle** property returns or sets a value associated with the degree of an ellipse's start angle in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Example

The following code example determines the size, shape, and appearance of an ellipse in CorelDRAW To determine the size of the ellipse, the Start Angle of the ellipse is set to 10 and the End Angle is set to 15. The new ellipse is created in a clockwise direction by setting the **Clockwise** property to True, and the ellipse type is set to a pie-type ellipse:

```
Sub EllipseStart()
With ActiveShape.Ellipse
    .StartAngle = 10
    .EndAngle = 15
    .Type = cdrPie
    .Clockwise = True
End With
End Sub
```

{button ,AL(`CLS_Ellipse;FNC_StartAngle')} **Related Topics**

Ellipse.EndAngle

Property **EndAngle** AS Double

Ellipse

Description

The **EndAngle** property returns or sets a value associated with the degree of an ellipse's end angle in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Example

The following code example determines the size, shape, and appearance of an ellipse in CorelDRAW To determine the size of the ellipse, the Start Angle of the ellipse is set to 10 and the End Angle is set to 15. The new ellipse is created in a clockwise direction by setting the **Clockwise** property to True, and the ellipse type is set to a pie-type ellipse:

```
Sub EllipseEnd()
With ActiveShape.Ellipse
    .StartAngle = 10
    .EndAngle = 15
    .Type = cdrPie
    .Clockwise = True
End With
End Sub
```

{button ,AL(`CLS_Ellipse;FNC_EndAngle')} **Related Topics**

Ellipse.Clockwise

Property **Clockwise** AS Boolean

Ellipse

Description

The **Clockwise** property returns or sets a True or False value indicating if the direction of the ellipse is a clockwise direction. If the value returned is True, the ellipse has a clockwise rotation.

An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object.

Example

The following code example determines the size, shape, and appearance of an ellipse in CorelDRAW To determine the size of the ellipse, the Start Angle of the ellipse is set to 10 and the End Angle is set to 15. The new ellipse is created in a clockwise direction by setting the **Clockwise** property to True, and the ellipse type is set to a pie-type ellipse:

```
Sub EllipseClockwise()
With ActiveShape.Ellipse
    .StartAngle = 10
    .EndAngle = 15
    .Type = cdrPie
    .Clockwise = True
End With
End Sub
```

{button ,AL(`CLS_Ellipse;FNC_Clockwise')} **Related Topics**

Polygon properties

Polygon Legend

- Sharpness
- Sides
- Type

Polygon

Class **Polygon**

Properties Referenced by

The **Polygon** class defines the characteristics of polygon objects and describes the look and behavior of the objects through its properties and methods. A polygon is a closed shape object with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

You can change the properties, such as number of points, of a polygon or star. You can also use the Shape tool or Pick tool to change the shape of the polygon or star.

If you want the changes you make to one node to be reflected on all associated nodes, you can mirror edit the polygon or star. When mirror editing, you can move, add, and remove segments and nodes. Nodes can be changed to smooth, cusp, or symmetrical, and segments can be straight or curved.

If you hold down SHIFT, CorelDRAW creates the polygon or star from the center outward as you drag.

{button ,AL(`CLS_Polygon')} **Related Topics**

cdrPolygon=0

cdrStar=1

cdrPolygonAsStar=2

Polygon.Type

Property **Type** AS **cdrPolygonType**

Polygon

Description

The **Type** property returns or sets a numerical value that identifies a polygon type in CorelDRAW. A polygon is a closed shape with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

The **Type** property returns a value of cdrPolygonType.

Example

The following code example displays the polygon type of the active shape in a message box:

```
Sub TypePolygon()
With ActiveShape.Polygon
   MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Polygon;FNC_Type')} **Related Topics**

Polygon.Sides

Property **Sides** AS Long

Polygon

Description

The **Sides** property returns or sets the number of sides within a polygon in CorelDRAW. A polygon is a closed shape with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

Example

The following code example displays the number of sides in active polygon shape in a message box:

```
Sub SidesPolygon()
With ActiveShape.Polygon
   MsgBox .Sides
End With
End Sub
```

{button ,AL(`CLS_Polygon;FNC_Sides')} **Related Topics**

Polygon.Sharpness

Property **Sharpness** AS Long

Polygon

Description

The **Sharpness** property returns or sets the sharpness of a polygon in CorelDRAW. A polygon is a closed shape with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

The **Sharpness** property defines the perspective of a polygon. Increased sharpness makes the polygon more defined in its appearance.

Example

The following code example sets the degree of sharpness in the active polygon shape in CorelDRAW:

```
Sub SharpPolygon()
With ActiveShape.Polygon
    .Sharpness = 25
End With
End Sub
```

{button ,AL(`CLS_Polygon;FNC_Sharpness')} **Related Topics**

Curve properties

Curve Legend

Closed

- Length

- Nodes

- Segments

- Subpaths

Curve methods

Curve Legend

- CreateSubPath
- IsOnCurve
- ReverseDirection
- Selection

Curve

Class **Curve**

Properties Methods Referenced by

The **Curve** class defines the characteristics of Curve objects and describes the look and behavior of the objects through its properties and methods.

A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the ConvertToCurves method of the Shape class.

{button ,AL(`CLS_Curve')} **Related Topics**

Curve.Length

Property **Length** AS Double

Curve

Description

The **Length** property returns the length of a selected curve object in CoreDRAW. Length is measured in document units, which may vary from document to document, and returns a value of cdrUnit.

The curve calculation is an approximation, not an absolute value. The precision of the calculation may be set with the CurvePrecision property of the Document object.

The **Length** property returns a Read-Only value.

Example

The following code example selects a shape object in the active document and determines if that shape is a curve object. If the shape is a curve, a message box displays the length of that curve, in a measurement of inches:

```
Sub CurveLength()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    MsgBox "The length of the curve is: " _
    & vbCrLf & s.Curve.Length & " inches"
End If
End Sub
```

{button ,AL(`CLS_Curve;FNC_Length')} **Related Topics**

cdrTenthMicron=0
cdrInch=1
cdrFoot=2
cdrMillimeter=3
cdrCentimeter=4
cdrPixel=5
cdrMile=6
cdrMeter=7
cdrKilometer=8
cdrDidots=9
cdrAgate=10
cdrYard=11
cdrPica=12
cdrCicero=13
cdrPoint=14

Curve.Subpaths

Property **Subpaths** AS **Subpaths**

Curve

Description

The **Subpaths** property returns a value associated with a curve object's subpath collection in CorelDRAW. A subpath is a curve or shape within a single curve object. Curve objects may have several subpaths and each member of the subpath collection is identified by its index number.

`Subpaths(2)` refers to the second subpath of a selected curve object in CorelDRAW.

The **Subpaths** property returns a Read-Only value.

Example

The following code example determines if the objects selected in the active document are curves and converts the non-curve objects in the selection to curves with the ConvertToCurves method. An ellipse is created around each curve's subpath and the horizontal and vertical coordinates of each subpath are used as the x and y coordinates of the ellipse:

```
Sub CurveSubPaths()
Dim s As Shape
Dim sp As SubPath
Dim x As Double, y As Double
For Each s In ActiveSelection.Shapes
    If s.Type <> cdrCurveShape Then
    s.ConvertToCurves
    End If
    If s.Type = cdrCurveShape Then
       For Each sp In s.Curve.Subpaths
          x = sp.PositionX
          y = sp.PositionY
          ActiveLayer.CreateEllipse2 x, y, 0.05
       Next sp
    End If
Next s
End Sub
```

{button ,AL(`CLS_Curve;FNC_Subpaths')} **Related Topics**

Curve.Nodes

Property **Nodes** AS **Nodes**

Curve

Description

The **Nodes** property returns a value associated with a curve object's Nodes collection in CorelDRAW. A node is a small square on a line, curve, or object outline used to edit the object. Curve objects may have numerous nodes, created along the curve's subpaths and each node is identified by its index number.

`Nodes(2).SubPathIndex` refers to the index number of the subpath containing the fourth node of a selected curve object in CorelDRAW.

The **Nodes** property returns a Read-Only value.

Example

The following code example selects a shape in the active document and determines if the selected shape is a curve object. If the shape is a curve, it displays the number of nodes found on the selected curve in a message box:

```
Sub CurveNodes()
Dim s As Shape
Dim nr As NodeRange
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    MsgBox "The selected shape has " & s.Curve.Nodes.Count & " nodes"
End If
End Sub
```

{button ,AL(`CLS_Curve;FNC_Nodes')} **Related Topics**

An index number is a reference to any collection that contains more than one object. It is used to identify each object in the collection. The index number can range from 1 to the number of available objects within the collection.

Curve.Segments

Property **Segments** AS **Segments**

Curve

Description

The **Segments** property returns a value associated with a curve object's Segments collection in CorelDRAW. A segment is the part of a curve that lies between two nodes. Segments are either curved or straight - both can be manipulated to change the appearance of the shape object, but straight segments must be converted to curved segments before they can be altered. Curve objects may have numerous segments along its subpaths and each segment is identified by its index number.

`Segments(1)` refers to the first segment of a selected curve object in CorelDRAW.

The **Segments** property returns a Read-Only value.

Example

The following code example makes a selection of shapes in the active document and determines if the objects selected are shape objects. It converts all non-curve objects in the selection to curves and changes the segment type for all of the subpath's segments to Linear segment types:

```
Sub CurveSegments()
Dim s As Shape
Dim seg As Segment
For Each s In ActiveSelection.Shapes
    If s.Type <> cdrCurveShape Then
    s.ConvertToCurves
    End If
    If s.Type = cdrCurveShape Then
        For Each seg In s.Curve.Segments
        seg.Type = cdrLineSegment
        Next seg
    End If
Next s
End Sub
```

{button ,AL(`CLS_Curve;FNC_Segments')} **Related Topics**

cdrLineSegment=0

cdrCurveSegment=1

cdrMixedSegments=2

Curve.Closed

Property **Closed** AS Boolean

Curve

Description

The **Closed** property returns or sets the value of a curve object's closed status in CorelDRAW. If a curve is closed (`Curve.Closed=True`), it is a self-contained graphic object. If a curve is open, it is a single or multi-segment shape that does not join all segments of the curve together. Curves are closed if all of its subpaths are closed.

A triangle resembles a closed curve object. The letter "V" resembles an open curve object.

The **Closed** property returns a Boolean value.

Example

The following code example selects an object in the active document and determines if the object is a curve object. If the selection is a curve, it changes its Closed status to True, joining all the segments in the shape to make a self-contained object. A color fountain fill is applied to the new closed curve object using the CreateColorEx method:

```
Sub CurveClosed()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    s.Curve.Closed = True
End If
s.Fill.ApplyFountainFill CreateColorEx(5005, 255, 0, 0), _
CreateColorEx(5005, 0, 0, 0), 1
End Sub
```

{button ,AL(`CLS_Curve;FNC_Closed')} **Related Topics**

cdrLinearFountainFill=1

cdrRadialFountainFill=2

cdrConicalFountainFill=3

cdrSquareFountainFill=4

Curve.Selection

Function **Selection**() AS **NodeRange**

Curve

Description

The **Selection** method returns or sets a value associated with a curve object's NodeRange in CorelDRAW. A Node Range contains a curve's pre-selected nodes and this property may only be used when the Shape tool is the active tool in CorelDRAW.

The **Selection** method returns NodeRange.

Example

The following code example selects an object in the active document and determines if the object is a curve object. If it is a curve, it evaluates the pre-selected nodes in the object's selection. If the node type in the selection is mixed nodes, it displays a message in a message box:

```
Sub CurveSelection()
Dim s As Shape
Dim nr As NodeRange
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
Set nr = s.Curve.Selection
If nf.type=cdrMixedNodes then
    MsgBox "The current curve selection has mixed nodes."
End If
End Sub
```

{button ,AL(`CLS_Curve;FNC_Selection')} **Related Topics**

cdrCuspNode=0

A cusp node is a node with control points that move independently from each other. A curve that passes through a cusp node can bend at a sharp angle.

cdrSmoothNode=1

A smooth node is a node with control points that are always opposite each other - when one moves, the other point moves as well. These control points may be different lengths and they produce a smooth transition between segments.

cdrSymmetricalNode=2

A symmetrical node is a node with control points that are always opposite each other, but unlike the smooth node, the control points of symmetrical nodes are always the same length. They produce the same curvature on both sides of the node.

cdrMixedNodes=3

A mixed node is a node with control points that are a combination of cusp nodes, smooth nodes, and symmetrical nodes.

Curve.CreateSubPath

Function **CreateSubPath**(ByVal **X** AS Double, ByVal **Y** AS Double) AS **SubPath**

Curve

Description

The **CreateSubPath** method creates or replaces a curve object's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. The CreateSubPath property requires an x and y coordinate to represent the first node in the subpath.

`Curve.CreateSubPath(1, 1)` creates a new subpath at the coordinates 1,1.

The **CreateSubPath** method returns SubPath.

Parameters	Description
X	Specifies the horizontal position (using the CorelDRAW rulers as a reference) for the first node in a curve object's new subpath. This horizontal coordinate is needed to place the new subpath in the active document.
Y	Specifies the vertical position (using the CorelDRAW rulers as a reference) for the first node in a curve object's new subpath. This vertical coordinate is needed to place the new subpath in the active document.

Example

The following code example creates a new curve object in the active document and creates a new subpath in that curve object. Using the coordinates (1, 1), the new subpath appears one inch to the right and one inch above the bottom left corner of the active page, set by the reference point.. Within that new subpath, two un-appended line segments are added with the AppendLineSegment method. The subpath is closed to create an inverted closed triangle:

```
Sub SubPathCreate()
Dim s As Shape
Dim sp As SubPath
Set s = ActiveLayer.CreateCurve
ActiveDocument.ReferencePoint = cdrBottomLeft
Set sp = s.Curve.CreateSubPath(1, 1)
    sp.AppendLineSegment False, 2, 2
    sp.AppendLineSegment False, 0, 2
    sp.Closed = True
End Sub
```

{button ,AL(`CLS_Curve;FNC_CreateSubPath')} **Related Topics**

CorelDRAW's VBA programming commands that specify locations on a page use coordinates as parameters. Coordinate values use the set document unit as the base unit of measurement and are expressed as being relative to the center of the current page, which has the coordinates (0,0).

Most CorelDRAW commands that use coordinates, such as the .**CreateSubPath** command (sets the position of the new subpath's first node), are affected by the application's current reference point, the point on the selected object's bounding box that the coordinates operate on. You can set the current reference point with the .ReferencePoint command.

cdrTopRight=1

cdrTopMiddle=2

cdrTopLeft=3

cdrMiddleLeft=4

cdrBottomLeft=5

cdrBottomMiddle=6

cdrBottomRight=7

cdrMiddleRight=8

cdrCenter=9

Curve.ReverseDirection

Sub **ReverseDirection**()

Curve

Description

The **ReverseDirection** method changes the direction of the flow for each curve object's subpath in CorelDRAW. If a curve object has a start and end position, the ReverseDirection method switches them so that the start position is now the end position within the curve object.

Example

The following code example creates a line object in the active document with the CreateLineSegment method. An arrowhead is applied to the start position of the line and the zoom of the active window is increases to 200%. The ReverseDirection method is applied and the arrowhead that was initially applied to a specific end of the line now appears at the opposite end of the line, because the start position has been switched with the end position in the line object:

```
Sub CurveReverse()
Dim s As Shape
Set s = ActiveLayer.CreateLineSegment(0, 0, 5, 5)
s.Outline.StartArrow = ArrowHeads(3)
ActiveWindow.ActiveView.Zoom = 200
s.Curve.ReverseDirection
End Sub
```

{button ,AL(`CLS_Curve;FNC_ReverseDirection')} **Related Topics**

Curve.IsOnCurve

Function **IsOnCurve**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal HotArea AS Double) AS **cdrPositionOfPointOverShape**

Curve

Description

The **IsOnCurve** method returns a value associated with a point (x and y coordinate) in relation to a curve object in CorelDRAW. A point may be inside, outside, or over the outline of a curve.

The IsOnCurve method returns cdrPositionOfPointOverShape.

Parameters	Description
X	Sets the horizontal position (using the CorelDRAW rulers as a reference) for a point in the active document, created in relation to the reference point.
Y	Sets the vertical position (using the CorelDRAW rulers as a reference) for a point in the active document, created in relation to the reference point.
HotArea	Defines a restricted area on the curve in which to evaluate the position of a point, in relation to the curve. This value is optional.

Example

The following code examples creates a rectangle in the active document with the CreateRectangle method and converts the rectangle to a curve. The ShowOnCurve function is called, coordinates are passed to the function, and a message box displays each point position in relation to the rectangle:

```
Sub OnCurve()
Dim s As Shape
Set s = ActiveLayer.CreateRectangle(0, 0, 2, 2)
s.ConvertToCurves
ShowOnCurve 1, 1, s.Curve.IsOnCurve(1, 1)
ShowOnCurve 1, 0, s.Curve.IsOnCurve(1, 0)
ShowOnCurve 1, 3, s.Curve.IsOnCurve(1, 3)
End Sub
```

{button ,AL(`CLS_Curve;FNC_IsOnCurve')} **Related Topics**

```
Private Function ShowOnCurve(x As Double, y As Double, v As cdrPositionOfPointOverShape)
Dim s As String
s = "Point (" & x & "," & y & ") is "
Select Case v
  Case cdrOutsideShape
    s = s & "outside the shape"
  Case cdrOnMarginOfShape
    s = s & "over the outline"
  Case cdrInsideShape
    s = s & "inside the shape"
End Select
MsgBox s
End Function
```

cdrOutsideShape=0

cdrOnMarginOfShape=1

cdrInsideShape=2

Page properties

Page Legend

- ActiveLayer

- Application
  - Background
  - Bleed
  - Color
- Guides
- Index
- Layers
  - Name
  - Orientation
- Paper
- Parent
  - PrintExportBackground
- Properties
  - Resolution
- Shapes
  - SizeHeight
  - SizeWidth

Page methods

Page Legend

Activate IDH_Page_Activate>main
CreateLayer IDH_Page_CreateLayer>main
Delete IDH_Page_Delete>main
FindShape IDH_Page_FindShape>main
FindShapes IDH_Page_FindShapes>main
GetSize IDH_Page_GetSize>main
MoveTo IDH_Page_MoveTo>main
SelectShapesAtPoint IDH_Page_SelectShapesAtPoint>main
SelectShapesFromRectangle IDH_Page_SelectShapesFromRectangle>main
SetSize IDH_Page_SetSize>main
TextFind IDH_Page_TextFind>main
TextReplace IDH_Page_TextReplace>main

UnlockAllShapes IDH_Page_UnlockAllShapes>main

Page

Class **Page**

Properties Methods Referenced by

The **Page** class defines the characteristics of page objects and describes the look and behavior of the objects through its properties and methods.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The default page orientation that you choose - landscape or portrait - automatically applies to every page in a multi-page document. Any new pages you add after creating the document are automatically assigned the same orientation.

However, you can change the orientation of an individual page within a multi-page document without affecting the orientation of other pages using the default.

CorelDRAW provides an array of preset page sizes to choose from, including standard North American and European sizes, and it also lets you create your own custom page sizes. In a multi-page document, the page size you assign anywhere in your document becomes the default size for all pages. Any new pages you add to the document later are automatically assigned the same default page size.

However, you can give individual pages within a multi-page document a custom size, without affecting other pages that are using the default.

{button ,AL(`CLS_Page')} **Related Topics**

Page.GetSize

Sub **GetSize**(ByRef **Width** As Double, ByRef **Height** As Double)

Description

The **GetSize** method gets the size of a page in CorelDRAW. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Parameters	Description
Width	The **Width** parameter sets the width of a page, measured in document units.
Height	The **Height** parameter sets the height of a page, measured in document units.

{button ,AL(`CLS_Page;FNC_GetSize')} **Related Topics**

Page.MoveTo

Sub **MoveTo**(ByVal **Index** As Long)

Description

The MoveTo method moves a page to a new position in a document in CorelDRAW. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Parameters	Description
Index	The **Index** parameter specifies the new index number of the moved page in the Pages collection of CorelDRAW. A new index number repositions the page within the collection.

{button ,AL(`CLS_Page;FNC_MoveTo')} **Related Topics**

Page.SetSize

Sub **SetSize**(ByVal **Width** As Double, ByVal **Height** As Double)

Description

The **SetSize** method sets the size of a page in CorelDRAW, by defining the height and width of the page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Parameters	Description
Width	The **Width** parameter sets the width of a page, measured in document units.
Height	The **Height** parameter sets the height of a page, measured in document units.

{button ,AL(`CLS_Page;FNC_SetSize')} **Related Topics**

Page.UnlockAllShapes

Sub **UnlockAllShapes**()

Description

The **UnlockAllShapes** method removes the lock from all shape object on a page in CorelDRAW. Once objects on the page are unlocked, editing of the objects is possible in CorelDRAW.

A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example unlocks all of the shape objects on the active page of CorelDRAW:

```
Sub CountProp()
ActivePage.UnlockAllShapes
End Sub
```

{button ,AL(`CLS_Page;FNC_UnlockAllShapes')} **Related Topics**

Page.Properties

Property **Properties** As **Properties**

Description

The **Properties** property returns a specified property of a page in a document by referencing the index number of a property. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Properties** property returns a Read-Only value.

Example

The following code example displays the number of properties associated with the active page in CorelDRAW:

```
Sub CountProp()
MsgBox ActivePage.Properties.Count
End Sub
```

{button ,AL(`CLS_Page;FNC_Properties')} **Related Topics**

Page.Name

Property **Name** AS String

Page

Description

The **Name** property returns or sets a string value that uniquely names a page in a document in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example displays the name of the active page in a message box:

```
Sub Test()
    MsgBox ActivePage.Name
End Sub
```

{button ,AL(`CLS_Page;FNC_Name')} **Related Topics**

Page.Application

Property **Application** AS **Application**

Page

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
With ActivePage
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Application')} **Related Topics**

Page.Parent

Property **Parent** AS **Pages**

Page

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the full file name of the active page's parent document:

```
Sub ParentName()
With ActivePage
    MsgBox .Parent.Parent.FullFileName
End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Parent')} **Related Topics**

Page.Layers

Property **Layers** AS **Layers**

Page

Description

The **Layers** property returns a value associated with the **Layers** collection on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain.

The **Layers** property returns a Read-Only value.

Example

The following code example changes the name of the first layer in the **Layers** collection and displays the new layer name in a message box:

```
Sub LayersItem()
With ActivePage.Layers
    .Item(1).Name = "CorelDRAW First Layer"
    MsgBox .Item(1).Name
End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Layers')} **Related Topics**

Page.Shapes

Property **Shapes** AS **Shapes**

Page

Description

The **Shapes** property returns a value associated with the **Shapes** collection on a page, across all layers, in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A shape is any object that can be displayed as several variations of a rectangle or a circle.

The **Layers** property returns a Read-Only value.

Example

The following code example counts the number of shape objects on the active page of CorelDRAW, across all layers:

```
Sub CountShape()
MsgBox ActivePage.Shapes.Count
End Sub
```

{button ,AL(`CLS_Page;FNC_Shapes')} **Related Topics**

Page.ActiveLayer

Property **ActiveLayer** AS **Layer**

Page

Description

The **ActiveLayer** property returns a value associated with the active layer on the active page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain.

The **ActiveLayer** property returns a Read-Only value.

Example

The following code example hides the active layer on the active page in CorelDRAW by setting its **Visible** property to False:

```
Sub HideLayer()
ActivePage.ActiveLayer.Visible = False
End Sub
```

{button ,AL(`CLS_Page;FNC_ActiveLayer')} **Related Topics**

Page.Paper

Property **Paper** AS String

Page

Description

The **Paper** property returns a string value associated with the paper size name in CorelDRAW. This value depends on the values set with the **SizeHeight** and **SizeWidth** properties of the **Page** class. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Paper** property returns a Read-Only value.

Example

The following code example checks to see if the active page size is set to "Letter" (8.5x10) size paper. If it is not set to Letter settings, the correct values are applied to the **SizeWidth** and **SizeHeight** values:

```
Sub Test()
    If ActivePage.Paper <> "Letter" Then
        ActivePage.SizeWidth = 8.5
        ActivePage.SizeHeight = 11
    End If
End Sub
```

{button ,AL(`CLS_Page;FNC_Paper')} **Related Topics**

Page.SizeWidth

Property **SizeWidth** AS Double

Page

Description

The **SizeWidth** property returns or sets the width size of a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example checks to see if the active page size is set to "Letter" (8.5x10) size paper. If it is not set to Letter settings, the correct values are applied to the **SizeWidth** and **SizeHeight** values:

```
Sub Test()
    If ActivePage.Paper <> "Letter" Then
        ActivePage.SizeWidth = 8.5
        ActivePage.SizeHeight = 11
    End If
End Sub
```

{button ,AL(`CLS_Page;FNC_SizeWidth')} **Related Topics**

Page.SizeHeight

Property **SizeHeight** AS Double

Page

Description

The **SizeHeight** property returns or sets the height size of a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example checks to see if the active page size is set to "Letter" (8.5x10) size paper. If it is not set to Letter settings, the correct values are applied to the **SizeWidth** and **SizeHeight** values:

```
Sub Test()
    If ActivePage.Paper <> "Letter" Then
        ActivePage.SizeWidth = 8.5
        ActivePage.SizeHeight = 11
    End If
End Sub
```

{button ,AL(`CLS_Page;FNC_SizeHeight')} **Related Topics**

Page.Resolution

Property **Resolution** AS Long

Page

Description

The **Resolution** property returns or sets the resolution of a page, measured in dots per inch (dpi), in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Resolution refers to the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size. The texture fill pattern is treated as a bitmap and its resolution determines the DPI value in the texture fill pattern.

DPI is a measure of a printer's resolution in dots per inch. Typical desktop laser printers print at 300 dpi. Image setters print at 1270 or 2540 dpi. Printers with higher dpi capabilities produce smoother and cleaner output. The term dpi is also used to measure scanning resolution and to indicate bitmap resolution.

Example

The following code example set the resolution of the active page to 400 dpi (dots per inch). The higher the resolution, the better your image will appear when printed:

```
Sub TextureResolution()
With ActivePage
    .Resolution = 400
End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Resolution')} **Related Topics**

Page.Bleed

Property **Bleed** AS Double

Page

Description

The **Bleed** property returns or sets a value associated with the bleed of an image on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Bleed refers to the part of the printed image that extends beyond the edge of the final page. The bleed ensures that the final image goes right to the edge of the paper after binding and trimming. A bleed is the extra bit of color, image or text that goes over the edge of the page so that after the printer cuts, binds, and trims it, there is no white space between the edges of your image and the edge of the paper. If you want a bleed, ensure that you draw your image so that it extends a few millimeters over the page borders. The bit extending over the borders is called the bleed, and CorelDRAW displays it with dotted lines along the page borders.

Example

The following code example sets the bleed value to 0.1 document units, document resolution to 150 dpi, and specifies yellow page background that should be omitted when printing or exporting the document:

```
Sub Test()
    With ActiveDocument.Pages(0)
        .Color.RGBAssign 255, 255, 0
        .PrintExportBackground = False
        .Bleed = 0.1
        .Resolution = 150
    End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Bleed')} **Related Topics**

Page.Orientation

Property **Orientation** AS **cdrPageOrientation**

Page

Description

The **Orientation** property returns or sets the orientation of a page in CorelDRAW. A page can be oriented as Landscape (width of the page is greater than the height) or Portrait (height of the page is greater than the width). A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The Orientation property returns a value of cdrPageOrientation.

Example

The following code example inserts a text object on each page, indicating the page orientation of the page in CorelDRAW:

```
Sub Test()
    Dim Orient As String
    Dim p As Page
    For Each p In ActiveDocument.Pages
        If p.Orientation = cdrPortrait Then
            Orient = "Portrait"
        Else
            Orient = "Landscape"
        End If
        p.ActiveLayer.CreateArtisticText 0, 0, Orient
    Next p
End Sub
```

{button ,AL(`CLS_Page;FNC_Orientation')} **Related Topics**

cdrPortrait=0

cdrLandscape=1

Page.Index

Property **Index** AS Long

Page

Description

The **Index** property returns the page number of a page in CorelDRAW. The index number uniquely identifies each page in a document. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Index** property returns a Read-Only value.

Example

The following code example displays the page number of the first page in the active document in a message box:

```
Sub Page()
With ActiveDocument
    MsgBox .Pages(1).Index
End With
End Sub
```

{button ,AL(`CLS_Page;FNC_Index')} **Related Topics**

Page.Activate

Sub **Activate**()

Page

Description

The **Activate** method opens a page in CorelDRAW and makes that page active in the current document. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example activates the first page of the active document in CorelDRAW and displays it on the screen:

```
Sub Test()
    ActiveDocument.Pages(1).Activate
End Sub
```

{button ,AL(`CLS_Page;FNC_Activate')} **Related Topics**

Page.Delete

Sub **Delete**()

Page

Description

The **Delete** method removes the active page from a document in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Deleting a page deletes all the shapes on that page in CorelDRAW.

Example

The following code example deletes all of the empty pages in the active document of CorelDRAW:

```
Sub Test()
    Dim p As Page
    For Each p In ActiveDocument.Pages
        If p.Shapes.Count = 0 Then p.Delete
    Next p
End Sub
```

{button ,AL(`CLS_Page;FNC_Delete')} **Related Topics**

Page.CreateLayer

Function **CreateLayer**(ByVal **LayerName** AS String) AS **Layer**

Page

Description

The **CreateLayer** method creates a new layer on the active page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateLayer** method, you must pass the **LayerName** value as a parameter:

Parameters	Description
LayerName	The **LayerName** parameter specifies a string value that uniquely identifies, and names, the new layer created on the active page with the **CreateLayer** method. "MyLayer" is an example of a **LayerName** value.

Example

The following code example adds a new layer named "MyLayer" to the active page, and creates a rectangle on the new layer:

```
Sub Test()
    Dim lr As Layer
    Set lr = ActivePage.CreateLayer("MyLayer")
    lr.CreateRectangle 0, 0, 2, 2
End Sub
```

{button ,AL(`CLS_Page;FNC_CreateLayer')} **Related Topics**

Page.TextFind

Function **TextFind**(ByVal **Text** AS String, ByVal **CaseSensitive** AS Boolean) AS **Shape**

Page

Description

The **TextFind** method finds and returns specific text on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

In order to use the **TextFind** method, you must pass the **Text** and **CaseSensitive** values as parameters:

Parameters	Description
Text	The **Text** parameter is a text string, contained in quotation marks, indicating the text to search for with the **TextFind** method. For example, "CorelDRAW Help" is a **Text** parameter.
CaseSensitive	The **CaseSensitive** parameter is a True or False value that indicates if the search string should look for case sensitive text. If the parameter is set to True, the text in the Text parameter must match the case of text in the text range.

Example

The following code example creates 3 text objects, finds the text object that contains the word "Sentence", and fills the text object red:

```
Sub Test()
    Dim s As Shape
    ActiveLayer.CreateArtisticText 0, 0, "Text1"
    ActiveLayer.CreateArtisticText 0, 3, "Some Sentence."
    ActiveLayer.CreateArtisticText 0, 5, "Word"
    Set s = ActivePage.TextFind("Sentence", True)
    If Not s Is Nothing Then
        s.Fill.UniformColor.RGBAssign 255, 0, 0
    End If
End Sub
```

{button ,AL(`CLS_Page;FNC_TextFind')} **Related Topics**

Page.TextReplace

Sub **TextReplace**(ByVal **OldText** AS String, ByVal **NewText** AS String, ByVal **CaseSensitive** AS Boolean, ByVal ReplaceSelectedOnly AS Boolean)

Page

Description

The **Replace** method searches for a specified text string on a page in CorelDRAW and replaces the text with a new text string. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

In order to search and locate specific text with the **TextReplace** method, you must pass the **OldText**, **NewText**, **CaseSensitive**, and **ReplaceSelectedOnly** values as parameters:

Parameters	Description
OldText	**OldText** is a text string in quotation marks, indicating the text in a text range that is replaced.
NewText	**NewText** is a text string in quotation marks, indicating the text that is added to a text range, in place of the text specified in the **OldText** parameter.
CaseSensitive	The **CaseSensitive** parameter is a True or False value that indicates if the search string should look for case sensitive text.
ReplaceSelectedOnly	The **ReplaceSelectedOnly** parameter is a True or False value that indicates if only the selected text is to be replaced with the **TextReplace** method. This value is optional and the default value is True.

Example

The following code example creates 3 text objects, and replaces the word "Some" with "Short", which results in one of the text objects becoming "Short Sentence.":

```
Sub Test()
    ActiveLayer.CreateArtisticText 0, 0, "Text1"
    ActiveLayer.CreateArtisticText 0, 3, "Some Sentence."
    ActiveLayer.CreateArtisticText 0, 5, "Word"
    ActivePage.TextReplace "Some", "Short", True, False
End Sub
```

{button ,AL(`CLS_Page;FNC_TextReplace')} **Related Topics**

Page.SelectShapesAtPoint

Function **SelectShapesAtPoint**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal **SelectUnfilled** AS Boolean, ByVal HotArea AS Double) AS **Shape**

Page

Description

The **SelectShapesAtPoint** method selects all the shapes beneath a given coordinate on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

This method creates a new selection and returns the selection shape object. In order to select a group of shape objects with this method, you must pass the several values as parameters:

Parameters	Description
X	The **X** parameter identifies the horizontal coordinate of the point on a page. All shapes below this point are selected with the **SelectShapesAtPoint** method. This value is measured in document units.
Y	The **Y** parameter identifies the vertical coordinate of the point on a page. All shapes below this point are selected with the **SelectShapesAtPoint** method. This value is measured in document units.
SelectUnfilled	Description of 'SelectUnfilled' goes here (**in**)
HotArea	Description of 'HotArea' goes here (**in**) Optional Default value = **-1**

Example

Example of usage goes here

{button ,AL(`CLS_Page;FNC_SelectShapesAtPoint')} **Related Topics**

Page.SelectShapesFromRectangle

Function **SelectShapesFromRectangle**(ByVal **x1** AS Double, ByVal **y1** AS Double, ByVal **x2** AS Double, ByVal **y2** AS Double, ByVal **Touch** AS Boolean) AS **Shape**

Page

Description

The **SelectShapesFromRectangle** method selects all the shapes within a rectangle shape on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

This method creates a new selection and returns the selection shape object. In order to select a group of shape objects with this method, you must pass the several values as parameters:

Parameters	Description
x1	The **x1** parameter defines the horizontal, or X, coordinate for the upper left corner of a rectangle shape on the active page. This value is measured in document units.
y1	The **y1** parameter defines the vertical, or Y, coordinate for the upper left corner of a rectangle shape on the active page. This value is measured in document units.
x2	The **x2** parameter defines the horizontal, or X, coordinate for the lower right corner of a rectangle shape on the active page. This value is measured in document units.
y2	The **y2** parameter defines the vertical, or Y, coordinate for the lower right corner of a rectangle shape on the active page. This value is measured in document units.
Touch	The **Touch** parameter returns a True or False value that specifies the inclusion of shapes marginally included in the rectangle shape on the active page. If Touch is True, the shapes not completely enclosed within the rectangle area are also selected.

Example

Example of usage goes here

{button ,AL(`CLS_Page;FNC_SelectShapesFromRectangle')} **Related Topics**

cdrPageBackgroundNone=0

cdrPageBackgroundSolid=1

cdrPageBackgroundBitmap=2

Page.Background

Property **Background** AS **cdrPageBackground**

Page

Description

The **Background** property returns or sets a value that indicates the type of background on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A background may contain a bitmap image, a solid fill, or no background at all. The Background property returns a value of cdrPageBackground.

Example

The following code example displays the background type of the active page in a message box:

```
Sub Test()
    Select Case ActiveDocument.Pages(0).Background
        Case cdrPageBackgroundNone
            MsgBox "No page background"
        Case cdrPageBackgroundSolid
            MsgBox "Solid color background"
        Case cdrPageBackgroundBitmap
            MsgBox "Bitmap page background"
    End Select
End Sub
```

{button ,AL(`CLS_Page;FNC_Background')} **Related Topics**

Page.Color

Property **Color** AS **Color**

Page

Description

The **Color** property returns or sets the color of a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Color** property only applies if the background type is "Solid".

Example

The following code example creates a red page background in the in the master page of the current document in CorelDRAW. Applying a red background to the master page applies that background to every page in the document:

```
Sub Test()
    ActiveDocument.Pages(0).Color.RGBAssign 255, 0, 0
End Sub
```

{button ,AL(`CLS_Page;FNC_Color')} **Related Topics**

Page.PrintExportBackground

Property **PrintExportBackground** AS Boolean

Page

Description

The **PrintExportBackground** property returns or sets a True or False value that indicates if the background of a page is included when printing or exporting that page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

Example

The following code example sets the bleed value to 0.1 document units, document resolution to 150 dpi, and specifies yellow page background that should be omitted when printing or exporting the document:

```
Sub Test()
    With ActiveDocument.Pages(0)
        .Color.RGBAssign 255, 255, 0
        .PrintExportBackground = False
        .Bleed = 0.1
        .Resolution = 150
    End With
End Sub
```

{button ,AL(`CLS_Page;FNC_PrintExportBackground')} **Related Topics**

Page.Guides

Property **Guides**(ByVal Type AS **cdrGuideType**) AS **ShapeRange**

Page

Description

The **Guides** property returns a value associated with the guidelines on a page in CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guidelines may be Horizontal, Vertical, or Slanted and any number of them may be created and saved with your drawing. Guidelines return a value of cdrGuideType. You can also enable snapping guidelines so that objects automatically align with the guidelines when moved or drawn nearby.

The **Guides** property returns a Read-Only value. You must pass the **Type** value as a parameter:

Parameters	Description
Type	The **Type** parameter specifies the guideline type on a page. This value returns cdrGuideType. This value is optional and the default value is cdrAllGuides.

Example

The following code example deletes all of the local vertical guidelines on the active page:

```
Sub Test()
    ActivePage.Guides(cdrVerticalGuide).Delete
End Sub
```

{button ,AL(`CLS_Page;FNC_Guides')} **Related Topics**

cdrAllGuides=-1

cdrHorizontalGuide=0

cdrVerticalGuide=1

cdrSlantedGuide=2

Page.FindShape

Function **FindShape**([ByVal Name As String], [ByVal Type As **cdrShapeType** = cdrNoShape (0)], [ByVal StaticID As Long = 0], [ByVal Recursive As Boolean = True]) As **Shape**

Page

Description

The **FindShape** method locates a shape object on a page in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **FindShape** method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter identifies the string value of the shape object that uniquely identifies the shape on the page. This value is optional.
Type	The **Type** parameter returns the shape type of the shape object on the page. This value returns cdrShapeType. This value is optional and the default value is cdrNoShape (0).

Example

Example of usage goes here

{button ,AL(`CLS_Page;FNC_FindShape')} **Related Topics**

Page.FindShapes

Function **FindShapes**([ByVal Name As String], [ByVal Type As **cdrShapeType** = cdrNoShape (0)], [ByVal Recursive As Boolean = True]) As **ShapeRange**

Page

Description

The **FindShapes** method locates all shape objects on a page in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **FindShape** method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter identifies the string value of the shape object that uniquely identifies the shape on the page. This value is optional.
Type	The **Type** parameter returns the shape type of the shape object on the page. This value returns cdrShapeType. This value is optional and the default value is cdrNoShape (0).

Example

Example of usage goes here

{button ,AL(`CLS_Page;FNC_FindShapes')} **Related Topics**

Layer properties

Layer Legend

- Application
  - Color
  - Editable
  - Master
- MasterProperties
  - Name
  - OverrideColor
- Parent
  - Printable
- Properties
- Shapes
  - Visible

Layer methods

Layer Legend

- Activate
- CreateArtisticText
- CreateConnector
- CreateCurve
- CreateCurveSegment
- CreateEllipse
- CreateEllipse2
- CreateFreeConnector
- CreateGridBoxes
- CreateGuide
- CreateGuideAngle
- CreateLineSegment
- CreateParagraphText
- CreatePolygon
- CreateRectangle
- CreateRectangle2
- CreateSpiral
- Delete
- FindShape
- FindShapes
- Import
- MoveAbove
- MoveBelow
- Paste

Layer

Class **Layer**

Properties Methods Referenced by

The **Layer** class defines the characteristics of layer objects and describes the look and behavior of the objects through its properties and methods.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The layering feature of CorelDRAW gives you added flexibility for organizing and editing the objects in your drawings. You can divide a drawing into multiple layers, each containing a portion of the drawing's contents. Together, layers act as a hierarchy that helps determine the vertical arrangement of a drawing's components. In this arrangement (the stacking order), objects on the top layer always overlay objects on the layer below.

The Layer Manager view lists all the layers in your document, without displaying sublevels or objects. It is the simplest view available, and is an easy way to switch between layers or move objects between layers.

You can assign a new name to any layer you create in the Object Manager. You can edit objects on any unlocked layer. You can also move and copy objects between any layers that are unlocked, as long as they are on the same page (or on the Master Page and another page).If you disable the Edit Across Layers function, you can work only on the active layer and the Desktop layer - you can't select or edit objects on inactive layers.

Each new file, regardless of how many pages it contains, has one master page. This master page controls four default layers: the Grid, Guides, and Desktop layers, plus one Layer (called Layer1) for drawing. To use a layer in the drawing, you must first make the layer active. Once active, a layer is ready to receive any new objects you draw, import, or paste onto it. In the Object Manager, the active layer appears highlighted in red. When you start a drawing, the default layer (Layer 1) is the active layer.

The Delete command removes the current layer highlighted in the Object Manager.

{button ,AL(`CLS_Layer')} **Related Topics**

Layer.CreateFreeConnector

Function **CreateFreeConnector**(ByVal **x1** As Double, ByVal **y1** As Double, ByVal **x2** As Double, ByVal **y2** As Double) As **Shape**

Description

The **CreateFreeConnector** method draws a connector line on a layer in CorelDRAW. A connector line is a straight line that connects two objects in a drawing. Connector lines can be set to remain attached to the nodes you choose or to shift to the closest nodes on the two connected objects. You can reposition a connector line only by moving the objects it is attached to.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Unlike normal connector lines that are attached to the nodes of objects, a free connector is an unbound connector line that can be unattached on a layer.

In order to use the **CreateFreeConnector** method, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter defines the horizontal, or X, coordinate for the start point of the new connector line on the layer. This value is measured in document units.
y1	The **y1** parameter defines the vertical, or Y, coordinate for the start point of the new connector line on the layer. This value is measured in document units.
x2	The **x2** parameter defines the horizontal, or X, coordinate for the end point of the new connector line on the layer. This value is measured in document units.
y2	The **y2** parameter defines the vertical, orY, coordinate for the end point of the new connector line on the layer. This value is measured in document units.

Example

The following code example creates an unbound connector line on the active layer in the current document:

```
Sub LayerConnector()
ActiveLayer.CreateFreeConnector 2, 3, 6, 7
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateFreeConnector')} **Related Topics**

Layer.MasterProperties

Property **MasterProperties** As **Properties**

Description

The **MasterProperties** property returns a value associated with the master properties of a layer in CorelDRAW. A master layer is a layer that contains information that you want to appear on every page of a multipage document. For example, you can use a master layer to place a header or footer on every page.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The **MasterProperties** property returns a Read-Only value.

Example

The following code example counts the number of properties associated with the active layer's master layer in CorelDRAW:

```
Sub Master()
MsgBox ActiveLayer.MasterProperties.Count
End Sub
```

{button ,AL(`CLS_Layer;FNC_MasterProperties')} **Related Topics**

Layer.Properties

Property **Properties** As **Properties**

Description

The **Properties** property returns a specified property of a layer by referencing the index number of a property. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The **Properties** property returns a Read-Only value.

Example

The following code example counts the number of properties associated with the active layer in CorelDRAW:

```
Sub Property()
MsgBox ActiveLayer.Properties.Count
End Sub
```

{button ,AL(`CLS_Layer;FNC_Properties')} **Related Topics**

Layer.Application

Property **Application** AS **Application**

Layer

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub LayerApp()
With ActiveLayer
   MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Layer;FNC_Application')} **Related Topics**

Layer.Parent

Property **Parent** AS **Layers**

Layer

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example counts the number of layer in the active layer's parent **Layers** collection in CorelDRAW:

```
Sub Layer()
MsgBox ActiveLayer.Parent.Count
End Sub
```

{button ,AL(`CLS_Layer;FNC_Parent')} **Related Topics**

Layer.Name

Property **Name** AS String

Layer

Description

The **Name** property returns or sets a string value that names a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Example

The following code example displays the name of the active layer in a message box:

```
Sub Test()
    MsgBox "Active Layer is " & ActiveLayer.Name
End Sub
```

{button ,AL(`CLS_Layer;FNC_Name')} **Related Topics**

Layer.Shapes

Property **Shapes** AS **Shapes**

Layer

Description

The **Shapes** property returns a collection of shape objects on a layer in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The **Shapes** property returns a Read-Only value.

Example

The following code example deletes the active layer and moves all shapes on the active layer to the next layer on the page:

```
Sub Test()
    Dim Target As Layer
    Dim s As Shape
    Dim idx As Long
    If ActivePage.Layers.Count = 1 Then
        MsgBox "Cannot delete a single layer"
    End If
    If ActiveLayer.Shapes.Count Then
        idx = 1 ' Determining the index of the active layer
        For Each Target In ActivePage.Layers
            If Target Is ActiveLayer Then Exit For
            idx = idx + 1
        Next Target
        ' Finding the index of target layer
        If idx = 1 Then idx = 2 Else idx = idx - 1
        Set Target = ActivePage.Layers(idx)
        ActiveDocument.ClearSelection
        ActiveLayer.Shapes.All.AddToSelection
        ActiveSelection.Layer = Target
    End If
    ActiveLayer.Delete
End Sub
```

{button ,AL(`CLS_Layer;FNC_Shapes')} **Related Topics**

Layer.Activate

Sub **Activate**()

Layer

Description

The **Activate** method opens a layer in CorelDRAW and makes that layer active in the current document. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Example

The following code example activates the first layer in the active page of CorelDRAW, making it the current layer:

```
Sub Test()
    ActivePage.Layers("Layer 1").Activate
End Sub
```

{button ,AL(`CLS_Layer;FNC_Activate')} **Related Topics**

Layer.Visible

Property **Visible** AS Boolean

Layer

Description

The **Visible** property returns a True or False value that indicates the visibility status of a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

If the **Visible** property is set the True, the layer is visible in CorelDRAW.

Example

The following code example makes the first layer in the **Layers** collection visible if its name is "Layer 1":

```
Sub LayerVisible()
With ActivePage.Layers
    If .Item(1).Name = "Layer 1" then
    .Item(1).Visible = True
End If
End With
End Sub
```

{button ,AL(`CLS_Layer;FNC_Visible')} **Related Topics**

Layer.Printable

Property **Printable** AS Boolean

Layer

Description

The **Printable** property returns or sets a True or False value that indicates the printability state of a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

CorelDRAW allows you to print selected layers of your drawing. If you enable a layer's print setting, the layer and its contents appear in printed copies of the drawing. If you disable a layer's print setting, the layer and its contents won't appear when you print the drawing.

If the **Printable** property of a layer is True, the layer and its contents are printable in CorelDRAW.

Example

The following code example makes the "Grid" layer on the master page printable:

```
Sub Test()
    ActiveDocument.Pages(0).Layers("Grid").Printable = True
End Sub
```

{button ,AL(`CLS_Layer;FNC_Printable')} **Related Topics**

Layer.Editable

Property **Editable** AS Boolean

Layer

Description

The **Editable** property returns or sets a True or False value that indicates the editing state of a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Locking a layer prevents accidental changes to its contents. When a layer is locked, objects on it can't be selected or edited. When you unlock the layer, you can change any object it contains as long as the object is not locked.

If the **Editable** property of a layer is True, the layer and its contents are printable in CorelDRAW.

Example

The following code example locks the active layer. Once the layer is locked, edits cannot be made to the layer or its contents:

```
Sub Test()
    ActiveLayer.Editable = False
End Sub
```

{button ,AL(`CLS_Layer;FNC_Editable')} **Related Topics**

Layer.Master

Property **Master** AS Boolean

Layer

Description

The **Master** property returns or sets a True or False value that indicates the master state of a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

A master layer is a layer that contains information that you want to appear on every page of a multi-page document. For example, you can use a master layer to place a header or footer on every page.

If the Master property is True, the layer acts as a master layer for the entire document and its contents will appear on every page in that document.

Example

The following code example makes the active layer a master layer, ensuring that its content appears on every page throughout the document:

```
Sub Test()
    ActiveLayer.Master = True
End Sub
```

{button ,AL(`CLS_Layer;FNC_Master')} **Related Topics**

Layer.OverrideColor

Property **OverrideColor** AS Boolean

Layer

Description

The **OverrideColor** property returns or sets a True or False value that indicates the enabled state of the override color feature on a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

You can display the contents of a selected layer as colored outlines. Changing to outlines, using color override, doesn't affect the objects' true appearance; it only affects the way they appear on-screen. This function is useful for identifying objects on different layers - for example, in a complex technical diagram - or even for changing the colors of the grid and guidelines.

Example

The following code example sets the color override feature to True in the active layer:

```
Sub Test()
    ActiveLayer.OverrideColor = True
End Sub
```

{button ,AL(`CLS_Layer;FNC_OverrideColor')} **Related Topics**

Layer.Color

Property **Color** AS **Color**

Layer

Description

The **Color** property returns or set the override color of a layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

You can display the contents of a selected layer as colored outlines. Changing to outlines, using color override, doesn't affect the objects' true appearance; it only affects the way they appear on-screen. This function is useful for identifying objects on different layers - for example, in a complex technical diagram - or even for changing the colors of the grid and guidelines.

Example

The following code example sets the override color of the active layer to RGB red:

```
Sub Test()
    ActiveLayer.Color.RGBAssign 255, 0, 0
End Sub
```

{button ,AL(`CLS_Layer;FNC_Color')} **Related Topics**

Layer.Delete

Sub **Delete**()

Layer

Description

The **Delete** method removes a layer, and its contents, from a document in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Example

The following code example deletes the first layer on the active page in CorelDRAW. Deleting the layer also deletes all of its contents:

```
Sub Test()
    ActivePage.Layers("Layer 1").Delete
End Sub
```

{button ,AL(`CLS_Layer;FNC_Delete')} **Related Topics**

Layer.MoveAbove

Sub **MoveAbove**(ByRef **Layer** AS **Layer**)

Layer

Description

The **MoveAbove** method places the current layer above a specified layer in CorelDRAW. The **MoveAbove** method changes the stacking order of a layer. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Stacking order is the vertical order of layers that help determine the appearance of the drawing. Each of these layers has its own internal stacking order.

The stacking order is most evident in drawings that contain overlapping objects with contrasting properties. If the objects do not overlap, the stacking order may not be evident. In all cases, however, the stacking order is determined by the order in which you add objects to the drawing. Until you rearrange them, the first object you draw occupies the lowest position, whereas the last object you draw occupies the top position.

You can change the stacking order of objects within any layer, sending them to the front or back, or behind or in front of another object.

In order to use the **MoveAbove** method, you must pass the **Layer** value as a parameter:

Parameters	Description
Layer	The **Layer** parameter identifies the layer object that has another layer moved above it with the **MoveAbove** method.

Example

The following code example moves the current layer and places it above layer "Layer 1":

```
Sub Test()
    ActiveLayer.MoveAbove ActivePage.Layers("Layer 1")
End Sub
```

{button ,AL(`CLS_Layer;FNC_MoveAbove')} **Related Topics**

Layer.MoveBelow

Sub **MoveBelow**(ByRef **Layer** AS **Layer**)

Layer

Description

The **MoveBelow** method places the current layer below a specified layer in CorelDRAW. The **MoveAbove** method changes the stacking order of a layer. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Stacking order is the vertical order of layers that help determine the appearance of the drawing. Each of these layers has its own internal stacking order.

The stacking order is most evident in drawings that contain overlapping objects with contrasting properties. If the objects do not overlap, the stacking order may not be evident. In all cases, however, the stacking order is determined by the order in which you add objects to the drawing. Until you rearrange them, the first object you draw occupies the lowest position, whereas the last object you draw occupies the top position.

You can change the stacking order of objects within any layer, sending them to the front or back, or behind or in front of another object.

In order to use the **MoveBelow** method, you must pass the **Layer** value as a parameter:

Parameters	Description
Layer	The **Layer** parameter identifies the layer object that has another layer moved below it with the **MoveBelow** method.

Example

The following code example moves the current layer and places it below layer "Layer 1":

```
Sub Test()
    ActiveLayer.MoveBelow ActivePage.Layers("Layer 1")
End Sub
```

{button ,AL(`CLS_Layer;FNC_MoveBelow')} **Related Topics**

Layer.Import

Sub **Import**(ByVal **FileName** AS String, ByVal Filter AS **cdrFilter**, ByVal MaintainLayersAndPages AS Boolean)

Layer

Description

The **Import** method imports shape objects from a disk file and places them onto the specified layer in CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **Import** method, you must pass the **FileName**, **Filter**, and **MaintainLayersAndPages** values as parameters:

Parameters	Description
FileName	The **FileName** parameter identifies the full path name of the disk file that contains the shapes that are imported into CorelDRAW with the **Import** method. For example, a CorelDRAW document named "Importing.cdr", stored on the root of the C drive, has a FileName value of "C:\Importing.cdr"
Filter	The **Filter** parameter returns or sets the filter that is used when importing objects in CorelDRAW. A filter is the name for an application that translates digital information from one form to another. Import/Export filters convert files from one format to another. This value returns cdrFilter. The **Filter** value is optional and the default value is cdrAutoSense (0).)
MaintainLayersAndPages	Enabling the Maintain Layers And Pages option lets you open the selected file and maintain the pages and layers contained in the file. The Corel .CMX file format gives you this option. This value is optional and the default value is False.

Example

The following code example imports the file "MyGraphic.cmx", stored on the root of C, into the active layer of CorelDRAW:

```
Sub Test()
    ActiveLayer.Import "C:\MyGraphic.cmx"
End Sub
```

{button ,AL(`CLS_Layer;FNC_Import')} **Related Topics**

Layer.CreateRectangle

Function **CreateRectangle**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal CornerUL AS Long, ByVal CornerUR AS Long, ByVal CornerLR AS Long, ByVal CornerLL AS Long) AS **Shape**

Layer

Description

The **CreateRectangle** method creates a rectangle on a layer in CorelDRAW. A rectangle is a geometric figure with four sides and four right angles, especially with adjacent sides unequal in length.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateRectangle** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter specifies the distance from the left side of the rectangle to the left side of the page frame in CorelDRAW. The value is measured in document units.
Top	The **Top** parameter specifies the distance from the top of the rectangle to the top of the page frame in CorelDRAW. The value is measured in document units.
Right	The **Right** parameter specifies the distance from the right side of the rectangle to the right side of the page frame in CorelDRAW. The value is measured in document units.
Bottom	The **Bottom** parameter specifies the distance from the bottom of the rectangle to the bottom of the page frame in CorelDRAW. The value is measured in document units.
CornerUL	The **CornerUL** parameter specifies the roundness measurement of the upper left corner of the rectangle. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. This value is optional and the default value is 0. Roundness values range between 0 and 100.
CornerUR	The **CornerUR** parameter specifies the roundness measurement of the upper right corner of the rectangle. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. This value is optional and the default value is 0. Roundness values range between 0 and 100.
CornerLR	The **CornerLR** parameter specifies the roundness measurement of the lower right corner of the rectangle. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. This value is optional and the default value is 0. Roundness values range between 0 and 100.
CornerLL	The **CornerLL** parameter specifies the roundness measurement of the lower left corner of the rectangle. When you round the corners of a rectangle, CorelDRAW splits each corner node in two and draws an arc between each of these two new nodes. This value is optional and the default value is 0. Roundness values range between 0 and 100.

Example

The following code example creates a rectangle and circle on the current page adds the rectangle and the circle to a shape range, and uses the **ToFitShapeRange** method to adjust the active view and zoom in so that the shapes in the shape range make up the entire view in CorelDRAW:

```
Sub ToFitShapeRange()
Dim s As Shape
Dim sr As New ShapeRange
Set s = ActiveLayer.CreateEllipse2(5, 5, 2)
sr.Add s       'Creates an ellipse and includes it in the defined range
Set s = ActiveLayer.CreateRectangle(0, 0, 2, 2)
sr.Add s       'Creates a rectangle and includes it in the defined range
```

```
ActiveWindow.ActiveView.ToFitShapeRange sr
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateRectangle')} **Related Topics**

Layer.CreateEllipse

Function **CreateEllipse**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal StartAngle AS Double, ByVal EndAngle AS Double, ByVal Arc AS Boolean) AS **Shape**

Layer

Description

The **CreateEllipse** method creates an ellipse on a layer in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object. The **CreateEllipse** method uses the corners of the ellipse bounding box to create the shape.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateEllipse** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter specifies the distance from the left side of the ellipse to the left side of the page frame in CorelDRAW. The value is measured in document units.
Top	The **Top** parameter specifies the distance from the top of the ellipse to the top of the page frame in CorelDRAW. The value is measured in document units.
Right	The **Right** parameter specifies the distance from the right side of the ellipse to the right side of the page frame in CorelDRAW. The value is measured in document units.
Bottom	The **Bottom** parameter specifies the distance from the bottom of the ellipse to the bottom of the page frame in CorelDRAW. The value is measured in document units.
StartAngle	The **StartAngle** parameter returns or sets a value associated with the degree of an ellipse's start angle. Altering the start angle changes the shape of the ellipse. This value is optional and the default value is 90.
EndAngle	The **EndAngle** parameter returns or sets a value associated with the degree of an ellipse's end angle. Altering the start angle changes the shape of the ellipse. This value is optional and the default value is 90.
Arc	The **Arc** parameter returns a True or False value that indicates if the newly created ellipse is changed into an arc. This value is optional and the default value is False.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.CMYKAssign 100, 0, 100, 0
'Fills the ellipse green
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateEllipse')} **Related Topics**

Layer.CreatePolygon

Function **CreatePolygon**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal **Sides** AS Long, ByVal Subpaths AS Long, ByVal Complexity AS Long, ByVal Star AS Boolean, ByVal StarComplexity AS Long, ByVal MaxComplexity AS Long) AS **Shape**

Layer

Description

The **CreatePolygon** method creates a new polygon shape object on a layer in CorelDRAW. A polygon is a closed shape object with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreatePolygon** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter specifies the distance from the left side of the polygon to the left side of the page frame in CorelDRAW. The value is measured in document units.
Top	The **Top** parameter specifies the distance from the top of the polygon to the top of the page frame in CorelDRAW. The value is measured in document units.
Right	The **Right** parameter specifies the distance from the right side of the polygon to the right side of the page frame in CorelDRAW. The value is measured in document units.
Bottom	The **Bottom** parameter specifies the distance from the bottom of the polygon to the bottom of the page frame in CorelDRAW. The value is measured in document units.
Sides	The **Sides** parameter specifies the number of sides in the polygon. A polygon can have 3 to 500 sides.
Subpaths	The **Subpaths** parameter specifies the number of subpaths in the polygon. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. This value is optional and the default value is 1.
Complexity	Description of 'Complexity' goes here (**in**) Optional Default value = **1**
Star	The **Star** parameter returns a True or False value that indicates if the polygon appears as a star shape on the layer. This value is optional and the default value is False.
StarComplexity	Description of 'StarComplexity' goes here (**in**) Optional Default value = **50**
MaxComplexity	Description of 'MaxComplexity' goes here (**in**) Optional Default value = **100**

Example

The following code example creates a star on the active layer of CorelDRAW:

```
Sub Test()
    ActiveLayer.CreatePolygon 0, 3, 3, 0, 5, 1, 1, True
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreatePolygon')} **Related Topics**

Layer.CreateGridBoxes

Function **CreateGridBoxes**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal **Wide** AS Long, ByVal **High** AS Long) AS **Shape**

Layer

Description

The **CreateGridBoxes** method creates a group of rectangles (as if using Graph Paper tool). on a layer in CorelDRAW. A rectangle is a geometric figure with four sides and four right angle, especially with adjacent sides unequal in length.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateGridBoxes** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter specifies the distance from the left side of the first rectangle grid box to the left side of the page frame in CorelDRAW. The value is measured in document units.
Top	The **Top** parameter specifies the distance from the top of the first rectangle grid box to the top of the page frame in CorelDRAW. The value is measured in document units.
Right	The **Right** parameter specifies the distance from the right side of the first rectangle grid box to the right side of the page frame in CorelDRAW. The value is measured in document units.
Bottom	The **Bottom** parameter specifies the distance from the bottom of the first rectangle grid box to the bottom of the page frame in CorelDRAW. The value is measured in document units.
Wide	The **Wide** parameter defines the horizontal measurement of each rectangle grid box. Its value is the width of the rectangle. This value is measured in document units.
High	The **High** parameter defines the vertical measurement of the rectangle. Its value is the height of the rectangle. This value is measured in document units.

Example

The following code example creates a series of rectangles on the active layer, resembling 2x3 graph paper:

```
Sub Test()
    ActiveLayer.CreateGridBoxes 0, 0, 5, 5, 2, 3
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateGridBoxes')} **Related Topics**

Layer.CreateSpiral

Function **CreateSpiral**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal **NumRevolutions** AS Long, ByVal **SpiralType** AS **cdrSpiralType**, ByVal **GrowthRate** AS Long) AS **Shape**

Layer

Description

The **CreateSpiral** method places a new spiral shape object on a layer in CorelDRAW. The Spiral tool lets you draw spiral shapes. There are two types of spirals: symmetrical and logarithmic. In a symmetrical spiral, the distance between each revolution of the spiral is constant. In a logarithmic spiral, this distance increases as the spiral progresses outward.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateSpiral** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter is a value that specifies the distance from the left side of the spiral to the left side of the page frame in CorelDRAW. It determines the position of the spiral object on the layer. The coordinate is measured in document units.
Top	The **Top** parameter is a value that specifies the distance from the top of the spiral to the top of the page frame in CorelDRAW. It determines the position of the spiral object on the layer. The coordinate is measured in document units.
Right	The **Right** parameter is a value that specifies the distance from the right side of the spiral to the right side of the page frame in CorelDRAW. It determines the position of the spiral object on the layer. The coordinate is measured in document units.
Bottom	The **Bottom** parameter is a value that specifies the distance from the bottom of the spiral to the bottom of the page frame in CorelDRAW. It determines the position of the spiral object on the layer. The coordinate is measured in document units.
NumRevolutions	The **NumRevolutions** parameter specifies the number of revolutions in the spiral. A revolution is a single rotation in the spiral object.
SpiralType	The **SpiralType** parameter specifies the type of spiral you create. A spiral can be symmetrical (distance between revolutions is constant) or logarithmic (distance between spirals increases as the spiral progresses outwards). This value returns cdrSpiralType.
GrowthRate	The **GrowthRate** parameter specifies the factor to increase the distance between revolutions in a logarithmic spiral. Values range from 1 to 100.

Example

The following code example creates a logarithmic spiral on the active layer of CorelDRAW. The spiral has 4 revolutions at a growth factor of 100 within each revolution:

```
Sub Test()
    ActiveLayer.CreateSpiral 0, 0, 5, 5, 4, cdrLogarithmic, 100
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateSpiral')} **Related Topics**

cdrSymmetric=0

cdrLogarithmic=1

Layer.CreateArtisticText

Function **CreateArtisticText**(ByVal **Left** AS Double, ByVal **Bottom** AS Double, ByVal **Text** AS String) AS **Shape**

Layer

Description

The **CreateArtisticText** method creates an artistic text object at a given point on a layer in CorelDRAW, with default font properties. Artistic text is created with the Text tool and is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateArtisticText** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter determines where to begin the horizontal positioning of the newly created artistic text. This value is measured in document units. The **Left** parameter combines with the **Bottom** parameter to create a coordinate that marks the start location of the new artistic text.
Bottom	The **Bottom** parameter determines where to begin the vertical positioning of the newly created artistic text. This value is measured in document units. The **Bottom** parameter combines with the **Left** parameter to create a coordinate that marks the start location of the new artistic text.
Text	The **Text** parameter is a string value that decides the content of the newly created artistic text. The value of the **Text** parameter becomes the actual text created with the **CreateArtisticText** method.

Example

The following code example creates a text string "Text" on the current layer of the active document at the lower left corner of the current page:

```
Sub Test()
    ActiveLayer.CreateArtisticText 0, 0, "Text"
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateArtisticText')} **Related Topics**

Layer.CreateParagraphText

Function **CreateParagraphText**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double, ByVal Text AS String) AS **Shape**

Layer

Description

The **CreateParagraphText** method creates a paragraph text object at a given point on a layer in CorelDRAW. Paragraph text is created in a paragraph text frame with the Text tool and is used to add large blocks of text for ads, brochures, and other text-intensive projects. A text frame is the rectangle that contains a block of Paragraph text created using the Text tool.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateParagraphText** method, you must pass several values as parameters:

Parameters	Description
Left	The **Left** parameter is a value that specifies the distance from the left side of the paragraph text frame to the left side of the page frame in CorelDRAW. The left coordinate is measured in document units.
Top	The **Top** parameter is a value that specifies the distance from the top side of the paragraph text frame to the top of the page frame in CorelDRAW. The top coordinate is measured in document units..
Right	The **Right** parameter is a value that specifies the distance from the right side of the paragraph text frame to the right side of the page frame in CorelDRAW. The right coordinate is measured in document units..
Bottom	The **Bottom** parameter is a value that specifies the distance from the bottom of the paragraph text frame to the bottom of the page frame in CorelDRAW. The bottom coordinate is measured in document units..
Text	The **Text** parameter is a string value that decides the content of the newly created paragraph text.

Example

The following code example creates a paragraph text object typed in bold Times New Roman font:

```
Sub Test()
    Dim s As Shape
    Set s = ActiveLayer.CreateParagraphText(0, 0, 4, 4, "Paragraph Text")
    s.Text.FontProperties.Name = "Times New Roman"
    s.Text.FontProperties.Style = cdrBoldFontStyle
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateParagraphText')} **Related Topics**

Layer.CreateCurveSegment

Function **CreateCurveSegment**(ByVal **StartX** AS Double, ByVal **StartY** AS Double, ByVal **EndX** AS Double, ByVal **EndY** AS Double, ByVal StartingControlPointLength AS Double, ByVal StartingControlPointAngle AS Double, ByVal EndingControlPointLength AS Double, ByVal EndingControlPointAngle AS Double) AS **Shape**

Layer

Description

The **CreateCurveSegment** method creates a single curve segment object on a layer in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself. A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the **ConvertToCurves** method of the **Shape** class.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateCurveSegment** method, you must pass several values as parameters:

Parameters	Description
StartX	The **StartX** parameter defines the horizontal, or X, coordinate for the starting point of the curve segment. This value is measured in document units.
StartY	The **StartY** parameter defines the vertical, or Y, coordinate for the starting point of the curve segment. This value is measured in document units..
EndX	The **EndX** parameter defines the horizontal, or X, coordinate for the ending point of the curve segment. This value is measured in document units..
EndY	The **EndY** parameter defines the vertical, orY, coordinate for the ending point of the curve segment. This value is measured in document units..
StartingControlPointLength	The **StartingControlPointLength** parameter specifies the length of the starting control point. The length is measured in document units. The length and angle of the control point dictate the appearance of the curve segment. This value is optional and the default value is -1.
StartingControlPointAngle	The **StartControlPointAngle** parameter specifies the angle, measured in degrees, of the starting control point. You can control the curve of a segment by varying the control point's angle. This value is optional and the default value is 0.
EndingControlPointLength	The **EndingControlPointLength** parameter specifies the length of the ending control point. The length is measured in document units. The length and angle of the control point dictate the appearance of the curve segment. This value is optional and the default value is -1.
EndingControlPointAngle	The **EndControlPointAngle** parameter specifies the angle, measured in degrees, of the ending control point. You can control the curve of a segment by varying the control point's angle. This value is optional and the default value is 0.

Example

The following code example creates a curve segment on the active layer with the **CreateCurveSegment** method. The starting point of the curve is (3, 3) and the ending point is (6, 6). The length of the starting and ending control points is 5 and the starting and ending angles of the control points are 45:

```
Sub CurveCreate()
ActiveLayer.CreateCurveSegment(3, 3, 6, 6, 5, 45, 5, 45)
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateCurveSegment')} **Related Topics**

Layer.CreateLineSegment

Function **CreateLineSegment**(ByVal **StartX** AS Double, ByVal **StartY** AS Double, ByVal **EndX** AS Double, ByVal **EndY** AS Double) AS **Shape**

Layer

Description

The **CreateLineSegment** method creates a line segment on a layer in CorelDRAW. A segment is a line or curve between nodes in a curve object. A line segment is created between to points on the layer. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateLineSegment** method, you must pass several values as parameters:

Parameters	Description
StartX	The **StartX** parameter defines the horizontal, or X, coordinate for the starting point of the line segment. This value is measured in document units.
StartY	The **StartY** parameter defines the vertical, or Y, coordinate for the starting point of the line segment. This value is measured in document units..
EndX	The **EndX** parameter defines the horizontal, or X, coordinate for the ending point of the line segment. This value is measured in document units..
EndY	The **EndY** parameter defines the vertical, orY, coordinate for the ending point of the line segment. This value is measured in document units..

Example

The following code example creates a new line in the active document with the **CreateLineSegment** method and creates a new subpath of the line with the **CreateSubPath** method. The subpath is not appended to the original line, leaving the lines to be treated as two separate objects. The **GetIntersections** method determines the intersection point where to two lines meet and the lines are broken at that crosspoint. That crosspoint now serves as the first node of the detached line segment that is offset, or moved slightly, from the other line segment:

```
Sub CrossPointOffset()
Dim s As Shape
Dim spath As SubPath
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s = ActiveLayer.CreateLineSegment(0, 0, 4, 4)
Set spath = s.Curve.CreateSubPath(0, 4)
    spath.AppendLineSegment False, 4, 0
Set cps = s.Curve.Segments(1).GetIntersections(s.Curve.Segments(2))
    s.Curve.Subpaths(1).BreakApartAt cps(1).Offset
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateLineSegment')} **Related Topics**

Layer.CreateConnector

Function **CreateConnector**(ByRef **Start** AS **ShapePoint**, ByRef **End** AS **ShapePoint**) AS **Shape**

Layer

Description

The **CreateConnector** method creates a connector line on a layer in CorelDRAW. A connector is a straight line that connects two objects in a drawing. Connector lines can be set to remain attached to the nodes you choose or to shift to the closest nodes on the two connected objects. You can reposition a connector line only by moving the its attached objects.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateConnector** method, you must pass the **Start** and **End** values as parameters:

Parameters	Description
Start	The **StartPoint** parameter returns or sets the starting point of the connector line object in CorelDRAW. This starting point is specified object's snap point. When you using use connector lines, special points on each object, called snap points, are activated. When a mouse passes over a snap point, it becomes visible. Objects can only be linked by connector lines at snap points, which are generally located at each node and at the center of the object.
End	The **EndPoint** parameter returns or sets the ending point of the connector line object in CorelDRAW. This ending point is a specified object's snap point. When you using use connector lines, special points on each object, called snap points, are activated. When a mouse passes over a snap point, it becomes visible. Objects can only be linked by connector lines at snap points, which are generally located at each node and at the center of the object.

Example

The following code example creates a rectangle and ellipse on the active layer, and a connector line is drawn between them, starting at the tenth snap point of the rectangle to the first snap point of the ellipse:

```
Sub PointEnd()
Dim s1 As Shape, s2 As Shape, c As Shape
Set s1 = ActiveLayer.CreateRectangle(0, 0, 3, 3)
Set s2 = ActiveLayer.CreateEllipse(4, 4, 6, 1)
Set c = ActiveLayer.CreateConnector(s1.Points(10), s2.Points(1))
    c.Connector.StartPoint = s1.Points(10)
    c.Connector.EndPoint = s2.Points(1)
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateConnector')} **Related Topics**

Layer.CreateCurve

Function **CreateCurve**() AS **Shape**

Layer

Description

The **CreateCurve** method creates an empty curve container on the active layer of CorelDRAW. A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the **ConvertToCurves** method of the **Shape** class.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

When you use the **CreateCurve** method, a real curve is not added to document until you add at least one segment to it using the **CreateSubpath**, **AppendLineSegment** or **AppendCurveSegment** commands.

Example

The following code example creates a new curve object on the active layer and creates a new subpath in that curve object. Using the coordinates (1, 1), the new subpath appears one inch to the right and one inch above the bottom left corner of the active page. Within that new subpath, two un-appended line segments are added with the **AppendLineSegment** method. The subpath is closed to create an inverted closed triangle:

```
Sub SubPathCreate()
Dim s As Shape
Dim sp As SubPath
Set s = ActiveLayer.CreateCurve
Set sp = s.Curve.CreateSubPath(1, 1)
    sp.AppendLineSegment False, 2, 2
    sp.AppendLineSegment False, 0, 2
    sp.Closed = True
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateCurve')} **Related Topics**

Layer.Paste

Function **Paste**() AS **Shape**

Layer

Description

The **Paste** method places the contents in the system clipboard on a layer in CorelDRAW. The clipboard is a temporary storage area that is used to hold cut or copied information. The clipboard is always present in Windows and it stores information until it is replaced by another object or selection that has been cut or copied, or until you quit the current session of Windows.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Example

The following code example copies the shapes from the current page to the clipboard, then adds 3 empty pages to the end of the document and pastes the objects on the active layer of each new page:

```
Sub Test()
    Dim p As Page
    Dim i As Long, idx As Long
    ActivePage.Shapes.All.Copy
    Set p = ActiveDocument.AddPages(3)
    idx = p.Index
    For i = idx To ActiveDocument.Pages.Count
        Set p = ActiveDocument.Pages(i)
        p.Activate
        p.ActiveLayer.Paste
    Next i
End Sub
```

{button ,AL(`CLS_Layer;FNC_Paste')} **Related Topics**

Layer.CreateGuideAngle

Function **CreateGuideAngle**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal **Angle** AS Double) AS **Shape**

Layer

Description

The **CreateGuideAngle** method creates a slanted guideline on the active layer in CorelDRAW, using a point and an angle. A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guidelines may be Horizontal, Vertical, or Slanted and any number of them may be created and saved with your drawing. Guidelines return a value of cdrGuideType. You can also enable snapping guidelines so that objects automatically align with the guidelines when moved or drawn nearby.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateGuideAngle** method, you must pass several values as parameters:

Parameters	Description
X	The **X** parameter defines the horizontal, or X, coordinate for the point that defines the slanted guideline. This value is measured in document units.
Y	The **Y** parameter defines the vertical, or Y, coordinate for the point that defines the slanted guideline. This value is measured in document units.
Angle	The **Angle** parameter defines the degree to which the guideline is slanted on a layer. The Angle value can range from 0 to 360

Example

The following code example creates a slanted guideline on the active layer with the coordinate (2, 2) and a slant measured at 45:

```
Sub Test()
ActiveLayer.CreateGuideAngle 2, 2, 45
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateGuideAngle')} **Related Topics**

Layer.CreateGuide

Function **CreateGuide**(ByVal **x1** AS Double, ByVal **y1** AS Double, ByVal **x2** AS Double, ByVal **y2** AS Double) AS **Shape**

Layer

Description

The **CreateGuide** method creates a guideline on the active layer in CorelDRAW, using two points. A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guidelines may be Horizontal, Vertical, or Slanted and any number of them may be created and saved with your drawing. Guidelines return a value of cdrGuideType. You can also enable snapping guidelines so that objects automatically align with the guidelines when moved or drawn nearby.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateGuide** method, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter defines the horizontal, or X, coordinate for the first point that defines the guideline. This value is measured in document units.
y1	The **y1** parameter defines the vertical, or Y, coordinate for the first point that defines the guideline. This value is measured in document units.
x2	The **x1** parameter defines the horizontal, or X, coordinate for the second point that defines the guideline. This value is measured in document units.
y2	The **y2** parameter defines the vertical, or Y, coordinate for the second point that defines the guideline. This value is measured in document units.

Example

The following code example creates a guideline on the active layer, between the coordinates (2, 2) and (5, 8)::

```
Sub Test()
ActiveLayer.CreateGuide 2, 2, 5, 8
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateGuide')} **Related Topics**

Layer.CreateEllipse2

Function **CreateEllipse2**(ByVal **CenterX** AS Double, ByVal **CenterY** AS Double, ByVal **Radius1** AS Double, ByVal Radius2 AS Double, ByVal StartAngle AS Double, ByVal EndAngle AS Double, ByVal Arc AS Boolean) AS **Shape**

Layer

Description

The **CreateEllipse2** method creates an ellipse on a layer in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is type of curve object. The **CreateEllipse2** method uses the coordinates of the shape center and horizontal (and vertical) radius to create the ellipse shape.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateEllipse2** method, you must pass several values as parameters:

Parameters	Description
CenterX	The **CenterX** parameter defines the horizontal, or X, coordinate that is the center point of the newly created ellipse. The value is measured in document units.
CenterY	The **CenterY** parameter defines the vertical, or Y, coordinate that is the center point of the newly created ellipse. The value is measured in document units.
Radius1	The **Radius1** parameter is the measurement from horizontal coordinate in the middle of the ellipse to circumference of its outer edge. The value is measured in document units.
Radius2	The **Radius2** parameter is the measurement from vertical coordinate in the middle of the ellipse to circumference of its outer edge. The value is measured in document units. This value is optional and the default value is 0.
StartAngle	The **StartAngle** parameter returns or sets a value associated with the degree of an ellipse's start angle. Altering the start angle changes the shape of the ellipse. This value is optional and the default value is 90.
EndAngle	The **EndAngle** parameter returns or sets a value associated with the degree of an ellipse's end angle. Altering the start angle changes the shape of the ellipse. This value is optional and the default value is 90.
Arc	The **Arc** parameter returns a True or False value that indicates if the newly created ellipse is changed into an arc. This value is optional and the default value is False.

Example

The following code example creates a circle in CorelDRAW and adjusts the active view to zoom in on the circle in the active document. The start and end angles are set to 25 degrees and the shape is not converted into an arc:

```
Sub ToFitShape()
Dim s As Shape
Set s = ActiveLayer.CreateEllipse2(5, 5, 2, 25, 25, False)
ActiveWindow.ActiveView.ToFitShape s
End Sub
```

{button ,AL(`CLS_Layer;FNC_CreateEllipse2')} **Related Topics**

Layer.FindShape

Function **FindShape**(ByVal Name AS String, ByVal Type AS **cdrShapeType**, ByVal StaticID As Long = 0, ByVal Recursive As Boolean = True) AS **Shape**

Layer

Description

The **FindShape** method locates a shape object on a layer in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **FindShape** method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter identifies the string value of the shape object that uniquely identifies the shape on the layer. This value is optional.
Type	The **Type** parameter returns the shape type of the shape object on the layer. This value returns cdrShapeType. This value is optional and the default value is cdrNoShape (0).

{button ,AL(`CLS_Layer;FNC_FindShape')} **Related Topics**

Layer.FindShapes

Function **FindShapes**(ByVal Name AS String, ByVal Type AS **cdrShapeType**, ByVal Recursive As Boolean = True) AS **ShapeRange**

Layer

Description

The **FindShapes** method locates all shape objects on a layer that meet the parameter values in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **FindShapes** method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter identifies the string value of the shape object that uniquely identifies the shape on the layer. This value is optional.
Type	The **Type** parameter returns the shape type of the shape object on the layer. This value returns cdrShapeType. This value is optional and the default value is cdrNoShape (0).

{button ,AL(`CLS_Layer;FNC_FindShapes')} **Related Topics**

Layer.CreateRectangle2

Function **CreateRectangle2**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal **Width** AS Double, ByVal **Height** AS Double, ByVal RadiusUL AS Double, ByVal RadiusUR AS Double, ByVal RadiusLR AS Double, ByVal RadiusLL AS Double) AS **Shape**

Layer

Description

The **CreateRectangle2** method creates a rectangle on a layer in CorelDRAW. A rectangle is a geometric figure with four sides and four right angles, especially with adjacent sides unequal in length. The **CreateRectangle2** method uses an upper left corner coordinate, width, height, and an actual corner radius value to create a rectangle on a layer.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

In order to use the **CreateRectangle2** method, you must pass several values as parameters:

Parameters	Description
X	The **X** parameter defines the horizontal, or X, position of the upper left corner coordinate of the rectangle. This value is measured in document units.
Y	The **Y** parameter defines the vertical, or Y, position of the upper left corner coordinate of the rectangle. This value is measured in document units.
Width	The **Width** parameter defines the horizontal measurement of the rectangle. Its value is the width of the rectangle. This value is measured in document units.
Height	The **Height** parameter defines the vertical measurement of the rectangle. Its value is the height of the rectangle. This value is measured in document units.
RadiusUL	The **RadiusUL** parameter specifies the actual corner radius measurement of the upper left corner of the rectangle. This value is optional and the default value is 0.
RadiusUR	The **RadiusUR** parameter specifies the actual corner radius measurement of the upper right corner of the rectangle. This value is optional and the default value is 0.
RadiusLR	The **RadiusLR** parameter specifies the actual corner radius measurement of the lower right corner of the rectangle. This value is optional and the default value is 0.
RadiusLL	The **RadiusLL** parameter specifies the actual corner radius measurement of the lower left corner of the rectangle. This value is optional and the default value is 0.

{button ,AL(`CLS_Layer;FNC_CreateRectangle2')} **Related Topics**

Shape properties

Shape Legend

- AbsoluteHScale
- AbsoluteSkew
- AbsoluteVScale
- Application
- Bitmap
- CloneLink
- Clones
- Connector
- Curve
  - DrapeFill
- Effect
- Effects
- Ellipse
  - Fill
- Guide
  - Layer
  - Locked
  - Name
- Next
- ObjectData
- OriginalHeight
- OriginalWidth
- Outline
  - OverprintFill
  - OverprintOutline
- Parent
- ParentGroup
- Points
- Polygon
  - PositionX
  - PositionY
- PowerClip
- PowerClipParent
- Previous
- Properties
- Rectangle
  - RotationAngle
  - RotationCenterX
  - RotationCenterY

- Selected
- Shapes
  - SizeHeight
  - SizeWidth
- StaticID
- Text
- Transparency
- Type
- URL

Shape methods

Shape Legend

- AddToPowerClip
- BreakApart
- ClearEffect
- Clone
- Combine
- ConvertToBitmap
- ConvertToBitmapEx
- ConvertToCurves
- Copy
- CreateArrowHead
- CreateBlend
- CreateContour
- CreateDropShadow
- CreateEnvelope
- CreateExtrude
- CreateLens
- CreatePerspective
- CreatePushPullDistortion
- CreateSelection
- CreateTwisterDistortion
- CreateZipperDistortion
- Cut
- Delete
- Duplicate
- Flip
- GetBoundingBox
- GetMatrix
- GetPosition
- GetSize
- Group
- Intersect
- IsOnShape
- Move
- OrderBackOf
- OrderBackOne
- OrderForwardOne
- OrderFrontOf
- OrderIsInFrontOf
- OrderReverse
- OrderToBack
- OrderToFront
- RemoveFromContainer
- Rotate
- RotateEx
- Separate
- SetBoundingBox
- SetMatrix
- SetPosition
- SetRotationCenter
- SetSize
- SetSizeEx

Skew
SkewEx
Stretch
StretchEx
Trim
Ungroup
UngroupAll
Weld

Shape.GetBoundingBox

Sub **GetBoundingBox**(ByRef **X** As Double, ByRef **Y** As Double, ByRef **Width** As Double, ByRef **Height** As Double, [ByVal UseOutline As Boolean = False])

Gets the shape bounding box relatively to its lower left corner

Member of **Shape**

Parameters	Description
X	Description of X goes here (**out**)
Y	Description of Y goes here (**out**)
Width	Description of Width goes here (**out**)
Height	Description of Height goes here (**out**)
UseOutline	Description of UseOutline goes here (**in**) Optional Default value = **False**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Shape;FNC_GetBoundingBox')} **Related Topics**

Shape.SetSizeEx

Sub **SetSizeEx**([ByVal Width As Double = 0], [ByVal Height As Double = 0], ByVal **CenterX** As Double, ByVal **CenterY** As Double)

Sets the shape size using the anchor point

Member of **Shape**

Parameters	Description
Width	Description of Width goes here (**in**) Optional Default value = **0**
Height	Description of Height goes here (**in**) Optional Default value = **0**
CenterX	Description of CenterX goes here (**in**)
CenterY	Description of CenterY goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Shape;FNC_SetSizeEx')} **Related Topics**

Shape

Class **Shape**

Properties Methods Referenced by

The **Shape** class defines the characteristics of Shape objects and describes the look and behavior of the objects through its properties and methods. A Shape is any object that can be displayed as several variations of a rectangle or a circle

Once you have opened and formatted your drawing page, you are ready to start drawing. To create your drawing, you will need to know how to create geometrical shapes, straight lines, curves, and irregular shapes.

Once you have created a shape or line, you can move and shape the object to create the effect you want. Some types of objects, such as rectangles and ellipses, can only be shaped in specific ways. For example, you can round one or more corners of a square or create a pie-shape or arc from a circle.

If you want to make more extensive changes to a rectangle, ellipse, or polygon, you can convert the object to a curve. For example, you can create a trapezoid from a rectangle by converting the rectangle to a curve object and then dragging the corners using the Shape tool.

CorelDRAW provides drawing tools for drawing basic shapes, such as rectangles, ellipses, polygons, stars, grids, and spirals. To draw a shape with one of these tools, drag diagonally in any direction until the shape is the size you want. For each tool, the Status Bar displays the dimensions of the shape as you draw it.

Once you have created basic objects, you can use a number of tools, buttons, and commands to change the shape of the object. For example, you can use the Shape tool to round the corners of a rectangle or shape an ellipse into an arc or a pie-shape.

The Shape tool lets you change the shape of all curve objects by editing their nodes and segments. You can also select and edit curves using the Pick tool.

{button ,AL(`CLS_Shape')} **Related Topics**

Shape.ClearEffect

Sub **ClearEffect**(ByVal **Type** As **cdrEffectType**)

Description

The **ClearEffect** method removes an effect from a shape object in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

Parameters	Description
Type	Specifies the effect to be removed in the shape object. This value returns cdrEffectType.

{button ,AL(`CLS_Shape;FNC_ClearEffect')} **Related Topics**

Shape.CreateSelection

Sub **CreateSelection**()

Description

The **CreateSelection** method makes a selection from a shape object in CorelDRAW.

{button ,AL(`CLS_Shape;FNC_CreateSelection')} **Related Topics**

Shape.RemoveFromContainer

Sub **RemoveFromContainer**([ByVal Level As Long = 0])

Description

The **RemoveFromContainer** method removes the active shape from a power clip object in CorelDRAW. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

When you PowerClip objects, you put one object inside another object or group of objects. One object becomes the contents, while the other becomes the container. You can create a container from any object you create using CorelDRAW, including shapes, lines, curves, Artistic text, and groups. A contents object can be any object you create using CorelDRAW or import from another program.

Parameters	Description
Level	If you created nested PowerClip objects and want to extract all contents objects in succession, you'll need to use perform this task for each nested level. This value is optional and the default value is 0.

{button ,AL(`CLS_Shape;FNC_RemoveFromContainer')} **Related Topics**

Shape.SetRotationCenter

Sub **SetRotationCenter**(ByVal **X** As Double, ByVal **Y** As Double)

Description

The **SetRotationCenter** method sets the center of rotation for a shape object in CorelDRAW. The center of rotation is the point around which an object rotates.

Parameters	Description
X	Sets the horizontal coordinate of the center of rotation.
Y	Sets the horizontal coordinate of the center of rotation.

{button ,AL(`CLS_Shape;FNC_SetRotationCenter')} **Related Topics**

Shape.StretchEx

Sub **StretchEx**(ByVal **StretchX** As Double, [ByVal StretchY As Double = 0], ByVal **CenterX** As Double, ByVal **CenterY** As Double, [ByVal StretchCharactersSize As Boolean = False])

Description

The **StretchEx** method stretches the shape using the anchor point of the shape in CorelDRAW. The anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object. The nine anchor points correspond to the eight handles on the object's Selection box and the center of the Selection box, which is represented by an X.

Parameters	Description
StretchX	Sets the value to stretch the shape horizontally.
StretchY	Sets the value to stretch the shape vertically. This value is optional and the default value is 0.
CenterX	Sets the x coordinate of the center of the shape object.
CenterY	Sets the y coordinate of the center of the shape object.

{button ,AL(`CLS_Shape;FNC_StretchEx')} **Related Topics**

Shape.Next

Property **Next**([ByVal Level As **cdrShapeLevel** = cdrLevelPage (3)], [ByVal EnterGroups As Boolean = False], [ByVal Loop As Boolean = True]) As **Shape**

Gets the next shape

Member of **Shape**

Read-Only

Parameters	Description
Level	Description of Level goes here (**in**) Optional Default value = **cdrLevelPage** (3)
EnterGroups	Description of EnterGroups goes here (**in**) Optional Default value = **False**
Loop	Description of Loop goes here (**in**) Optional Default value = **True**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Shape;FNC_Next')} **Related Topics**

Shape.Previous

Property **Previous**([ByVal Level As **cdrShapeLevel** = cdrLevelPage (3)], [ByVal EnterGroups As Boolean = False], [ByVal Loop As Boolean = True]) As **Shape**

Gets the previous shape

Member of **Shape**

Read-Only

Parameters	Description
Level	Description of Level goes here (**in**) Optional Default value = **cdrLevelPage** (3)
EnterGroups	Description of EnterGroups goes here (**in**) Optional Default value = **False**
Loop	Description of Loop goes here (**in**) Optional Default value = **True**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Shape;FNC_Previous')} **Related Topics**

Shape.Application

Property **Application** AS **Application**

Shape

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub Test()
MsgBox ActiveShape.Application.Version
End Sub
```

{button ,AL(`CLS_Shape;FNC_Application')} **Related Topics**

Shape.Parent

Property **Parent** AS Object

Shape

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the active shape's parent layer in CorelDRAW:

```
Sub Test()
MsgBox ActiveShape.Parent.Name
End Sub
```

{button ,AL(`CLS_Shape;FNC_Parent')} **Related Topics**

Shape.StaticID

Property **StaticID** AS Long

Shape

Description

The **StaticID** property returns the object data manager's static id reference to a shape object in CorelDRAW. The Object Manager displays the vertical structure - the stacking order - of objects, layers, and pages in your document. For each object, the Object Manager displays a small icon and a brief description indicating its basic fill and outline properties. You can set the display options for the pages, layers, and objects using the Object Manager.

This property returns a Read-Only value.

Example

The following code example displays the static ID of the active shape in CorelDRAW:

```
Sub Test()
MsgBox ActiveShape.StaticID
End Sub
```

{button ,AL(`CLS_Shape;FNC_StaticID')} **Related Topics**

Shape.ConvertToCurves

Sub **ConvertToCurves**()

Shape

Description

The **ConvertToCurves** method converts a shape object to a curve object in CorelDRAW. A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the ConvertToCurves method of the Shape class.

Example

The following code example makes a selection of shapes in the active document and determines if the objects selected are shape objects. It converts all non-curve objects in the selection to curves and changes the segment type for all of the subpath's segments to Linear segment types:

```
Sub CurveSegments()
Dim s As Shape
Dim seg As Segment
For Each s In ActiveSelection.Shapes
    If s.Type <> cdrCurveShape Then
    s.ConvertToCurves
    End If
    If s.Type = cdrCurveShape Then
        For Each seg In s.Curve.Segments
        seg.Type = cdrLineSegment
        Next seg
    End If
Next s
End Sub
```

{button ,AL(`CLS_Shape;FNC_ConvertToCurves')} **Related Topics**

Shape.Name

Property **Name** AS String

Shape

Description

The **Name** property returns or sets a string value that names a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the name the active shape in CorelDRAW:

```
Sub Test()
MsgBox ActiveShape.Name
End Sub
```

{button ,AL(`CLS_Shape;FNC_Name')} **Related Topics**

Shape.Shapes

Property **Shapes** AS **Shapes**

Shape

Description

The **Shapes** property returns a collection of shape objects on a layer in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

This property returns a Read-Only value.

{button ,AL(`CLS_Shape;FNC_Shapes')} **Related Topics**

Shape.Rectangle

Property **Rectangle** AS **Rectangle**

Shape

Description

The **Rectangle** property returns a value associated with a rectangle's properties in a shape object in CorelDRAW. A rectangle is a geometric figure with four sides and four right angles, especially with adjacent sides unequal in length.

This property returns a Read-Only value.

Example

The following code example sets varying degrees of roundness for each corner in the active rectangle shape in CorelDRAW. A message box displays the EqualCorners property, which is false since all corners have been set with different values:

```
Sub Rec()
With ActiveShape.Rectangle
    .CornerUpperLeft = 50
    .CornerUpperRight = 65
    .CornerLowerLeft = 90
    .CornerLowerRight = 25
    MsgBox .EqualCorners
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Rectangle')} **Related Topics**

Shape.PositionX

Property **PositionX** AS Double

Shape

Description

The **PositionX** property returns or sets the horizontal coordinate of a shape on a page, according to the document's reference point in CorelDRAW.

A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the X and Y coordinates of the active shape in CorelDRAW:

```
Sub Position()
With ActiveShape
    MsgBox .PositionX
    MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_PositionX')} **Related Topics**

Shape.PositionY

Property **PositionY** AS Double

Shape

Description

The **PositionY** property returns or sets the vertical coordinate of a shape on a page, according to the document's reference point in CorelDRAW.

A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the X and Y coordinates of the active shape in CorelDRAW:

```
Sub Position()
With ActiveShape
    MsgBox .PositionX
    MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_PositionY')} **Related Topics**

Shape.SizeWidth

Property **SizeWidth** AS Double

Shape

Description

The **SizeWidth** property returns or set the width of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the width and height of the active shape in CorelDRAW:

```
Sub Position()
With ActiveShape
    MsgBox .SizeHeight
    MsgBox .SizeWidth
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_SizeWidth')} **Related Topics**

Shape.SizeHeight

Property **SizeHeight** AS Double

Shape

Description

The **SizeHeight** property returns or set the height of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the width and height of the active shape in CorelDRAW:

```
Sub Position()
With ActiveShape
    MsgBox .SizeHeight
    MsgBox .SizeWidth
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_SizeHeight')} **Related Topics**

Shape.Ellipse

Property **Ellipse** AS **Ellipse**

Shape

Description

The **Ellipse** property returns a value associated with an ellipse's properties in a shape object in CorelDRAW. An ellipse is created with the Ellipse Tool. In its simplest form, an ellipse is a variation of a circular shape object. An ellipse is a closed plane curve generated by a point moving in such a way that the sums of its distances from two fixed points are constant. An ellipse is a type of curve object.

This property returns a Read-Only value.

Example

The following code example determines the size, shape, and appearance of an ellipse in CorelDRAW To determine the size of the ellipse, the Start Angle of the ellipse is set to 10 and the End Angle is set to 15. The new ellipse is created in a clockwise direction by setting the Clockwise property to True, and the ellipse type is set to a pie-type ellipse:

```
Sub EllipseType()
With ActiveShape.Ellipse
    .StartAngle = 10
    .EndAngle = 15
    .Type = cdrPie
    .Clockwise = True
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Ellipse')} **Related Topics**

Shape.Polygon

Property **Polygon** AS **Polygon**

Shape

Description

The **Polygon** property returns a value associated with an polygon's properties in a shape object in CorelDRAW. A polygon is a closed shape object with three to 500 sides. In CorelDRAW, you can create simple polygons (e.g., pentagons) or complex, multisided polygons (e.g., stars) using the Polygon tool.

This property returns a Read-Only value.

Example

The following code example displays the polygon type of the active shape in a message box:

```
Sub TypePolygon()
With ActiveShape.Polygon
   MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Polygon')} **Related Topics**

Shape.Curve

Property **Curve** AS **Curve**

Shape

Description

The **Curve** property returns a value associated with a curve's properties in a shape object in CorelDRAW. A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the **ConvertToCurves** method.

This property returns a Read-Only value.

Example

The following code example selects a shape object in the active document and determines if that shape is a curve object. If the shape is a curve, a message box displays the length of that curve, in a measurement of inches:

```
Sub CurveLength()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    MsgBox "The length of the curve is: " _
    & vbCrLf & s.Curve.Length & " inches"
End If
End Sub
```

{button ,AL(`CLS_Shape;FNC_Curve')} **Related Topics**

Shape.Bitmap

Property **Bitmap** AS **Bitmap**

Shape

Description

The **Bitmap** property returns a value associated with a bitmap's properties in a shape object in CorelDRAW. A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

This property returns a Read-Only value.

Example

The following code example displays the width and height of a bitmap object in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    MsgBox .SizeWidth
    MsgBox .SizeHeight
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Bitmap')} **Related Topics**

Shape.Type

Property **Type** AS **cdrShapeType**

Shape

Description

The **Type** property returns the shape type of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

This property is Read-Only and returns a value of cdrShapeType.

Example

The following code example displays the shape type of the active shape in CorelDRAW:

```
Sub Test()
MsgBox ActiveShape.Type
End Sub
```

{button ,AL(`CLS_Shape;FNC_Type')} **Related Topics**

Shape.Outline

Property **Outline** AS **Outline**

Shape

Description

The **Outline** property returns a value associated with an outline's properties in a shape object in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

This property returns a Read-Only value.

Example

The following code example displays the name of the outline color in the active shape of CorelDRAW:

```
Sub Color()
MsgBox ActiveShape.Outline.Color.Name
End Sub
```

{button ,AL(`CLS_Shape;FNC_Outline')} **Related Topics**

Shape.Fill

Property **Fill** AS **Fill**

Shape

Description

The **Fill** property returns or sets the fill properties of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

Example

The following code example set the fill type in the active shape of CorelDRAW. The fill is a fountain fill:

```
Sub FillType()
With ActiveShape.Fill
.Type = cdrFountainFill
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Fill')} **Related Topics**

Shape.Text

Property **Text** AS **Text**

Shape

Description

The **Text** property returns a value associated with a text object's properties in a shape object in CorelDRAW. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool.

Clicking in the Drawing Window and typing creates Artistic text. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. An Artistic text object can contain up to 32,000 characters. CorelDRAW automatically applies the default Artistic text style, which you can change using the Styles Manager.

This property returns a Read-Only value.

Example

The following code example displays the number of characters in the text string "CorelDRAW Help", using the **CharacterCount** property:

```
Sub TextCount()
With ActiveShape.Text
    MsgBox .CharacterCount
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Text')} **Related Topics**

Shape.Delete

Sub **Delete**()

Shape

Description

The **Delete** method deletes a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example deletes the active shape in CorelDRAW:

```
Sub Test()
ActiveShape.Delete
End Sub
```

{button ,AL(`CLS_Shape;FNC_Delete')} **Related Topics**

Shape.Duplicate

Function **Duplicate**(ByVal OffsetX AS Double, ByVal OffsetY AS Double) AS **Shape**

Shape

Description

The **Duplicate** method duplicates a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
OffsetX	The **OffsetX** parameter specifies the horizontal distance to offset the duplicated shape object. This value is optional and the default value is 0.
OffsetY	The **OffsetY** parameter specifies the vertical distance to offset the duplicated shape object. This value is optional and the default value is 0.

Example

The following code example duplicates the active shape in CorelDRAW:

```
Sub Test()
ActiveShape.Duplicate
End Sub
```

{button ,AL(`CLS_Shape;FNC_Duplicate')} **Related Topics**

Shape.Skew

Sub **Skew**(ByVal **AngleX** AS Double, ByVal **AngleY** AS Double)

Shape

Description

The **Skew** method skews a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

Parameters	Description
AngleX	The **AngleX** parameter specifies the degree in which to slant the shape object horizontally.
AngleY	The **AngleY** parameter specifies the degree in which to slant the shape object vertically.

Example

The following code example skews the active shape in CorelDRAW:

```
Sub Test()
ActiveShape.Skew (2, 3)
End Sub
```

{button ,AL(`CLS_Shape;FNC_Skew')} **Related Topics**

Shape.Move

Sub **Move**(ByVal **DeltaX** AS Double, ByVal **DeltaY** AS Double)

Shape

Description

The **Move** method moves a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
DeltaX	The **DeltaX** parameter specifies the horizontal distance to move a shape with the **Move** method. Delta refers to a limited incremental value in a variable.
DeltaY	The **DeltaY** parameter specifies the vertical distance to move a shape with the **Move** method. Delta refers to a limited incremental value in a variable.

Example

The following code example move the active shape up and over 1 in CorelDRAW:

```
Sub Test()
ActiveShape.Move (1, 1)
End Sub
```

{button ,AL(`CLS_Shape;FNC_Move')} **Related Topics**

Shape.RotationAngle

Property **RotationAngle** AS Double

Shape

Description

The **RotationAngle** property returns or set the rotation angle of a shape object on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

If `.RotationAngle = 45`, the shape rotates by 45 degrees on a page in CorelDRAW.

Example

The following code example sets the rotation angle for the active shape:

```
Sub Test()
With ActiveShape
    .RotationAngle = 45
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_RotationAngle')} **Related Topics**

Shape.RotationCenterX

Property **RotationCenterX** AS Double

Shape

Description

The **RotationCenterX** returns or sets the horizontal coordinate for the center of rotation of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The center of rotation is the point around which an object rotates.

Example

The following code example sets the center or rotation in the active shape to (2,2):

```
Sub Test()
With ActiveShape
    .RotationCenterX = 2
    .RotationCenterY = 2
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_RotationCenterX')} **Related Topics**

Shape.RotationCenterY

Property **RotationCenterY** AS Double

Shape

Description

The **RotationCenterY** returns or sets the vertical coordinate for the center of rotation of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The center of rotation is the point around which an object rotates.

Example

The following code example sets the center or rotation in the active shape to (2,2):

```
Sub Test()
With ActiveShape
    .RotationCenterX = 2
    .RotationCenterY = 2
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_RotationCenterY')} **Related Topics**

Shape.Rotate

Sub **Rotate**(ByVal **Angle** AS Double)

Shape

Description

The **Rotate** method rotates a shape object a specified degree in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Angle	The **Angle** parameter specifies, in degrees, the amount to rotate a shape object.

Example

The following code example rotates the active shape 45 degrees:

```
Sub Test()
ActiveShape.Rotate (45)
End Sub
```

{button ,AL(`CLS_Shape;FNC_Rotate')} **Related Topics**

Shape.ConvertToBitmap

Function **ConvertToBitmap**(ByVal BitDepth AS Long, ByVal Grayscale AS Boolean, ByVal Dithered AS Boolean, ByVal TransparentBG AS Boolean, ByVal Resolution AS Long, ByVal AntiAliasing AS **cdrAntiAliasingType**, ByVal UseColorProfile AS Boolean) AS **Shape**

Shape

Description

The **ConvertToBitmap** method converts a shape object from a vector image to a bitmap image in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Parameters	Description
BitDepth	Lets you specify bit depth. This value is optional and the default value is 24.
Grayscale	Set to TRUE (-1) converts to grayscale. This value is optional and the default value is False.
Dithered	Set to TRUE (-1) enables dithering. This value is optional and the default value is True.
TransparentBG	Set to TRUE (-1) enables transparent background. This value is optional and the default value is True.
Resolution	Lets you specify the resolution. This value is optional and the default value is 72.
AntiAliasing	Lets you specify the anti aliasing type.This value is optional and returns cdrAntiAliasingType. The default value is cdrNormalAntiAliasing.
UseColorProfile	Set to TRUE (-1) use the color profile. This value is optional and the default value is True.

{button ,AL(`CLS_Shape;FNC_ConvertToBitmap')} **Related Topics**

Shape.Group

Function **Group**() AS **Shape**

Shape

Description

The **Group** method creates a group with the active shape, only if the shape is a selection of shapes in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_Shape;FNC_Group')} **Related Topics**

Shape.Ungroup

Sub **Ungroup**()

Shape

Description

The **Ungroup** method removes the grouping feature from a selection of shape objects in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_Shape;FNC_Ungroup')} **Related Topics**

Shape.UngroupAll

Sub **UngroupAll**()

Shape

Description

The **UngroupAll** method removes the grouping feature from all nested groups of shape objects in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_Shape;FNC_UngroupAll')} **Related Topics**

Shape.OrderToFront

Sub **OrderToFront**()

Shape

Description

The **OrderToFront** method arranges the stacking order of a shape object by moving it to the front of the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_Shape;FNC_OrderToFront')} **Related Topics**

Shape.OrderToBack

Sub **OrderToBack**()

Shape

Description

The **OrderToBack** method arranges the stacking order of a shape object by moving it to the back of the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_Shape;FNC_OrderToBack')} **Related Topics**

Shape.OrderForwardOne

Sub **OrderForwardOne**()

Shape

Description

The **OrderForwardOne** method arranges the stacking order of a shape object by moving it forward one in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_Shape;FNC_OrderForwardOne')} **Related Topics**

Shape.OrderBackOne

Sub **OrderBackOne**()

Shape

Description

The **OrderBackOne** method arranges the stacking order of a shape object by moving it back one in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_Shape;FNC_OrderBackOne')} **Related Topics**

Shape.OrderFrontOf

Sub **OrderFrontOf**(ByRef **Shape** AS **Shape**)

Shape

Description

The **OrderFrontOf** method arranges the stacking order of a shape object by moving it in front of a specified shape in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

Parameters	Description
Shape	The **Shape** parameter identifies the Shape object that will have another shape placed in front of it in the stacking order of CorelDRAW.

{button ,AL(`CLS_Shape;FNC_OrderFrontOf')} **Related Topics**

Shape.OrderBackOf

Sub **OrderBackOf**(ByRef **Shape** AS **Shape**)

Shape

Description

The **OrderBackOf** method arranges the stacking order of a shape object by moving it in back of a specified shape in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

Parameters	Description
Shape	The **Shape** parameter identifies the Shape object that will have another shape placed in back of it in the stacking order of CorelDRAW.

{button ,AL(`CLS_Shape;FNC_OrderBackOf')} **Related Topics**

Shape.OrderIsInFrontOf

Function **OrderIsInFrontOf**(ByRef **Shape** AS **Shape**) AS Boolean

Shape

Description

The **OrderIsInFrontOf** method returns a True or False value that determines whether or not one shape is in front of the other in the stacking order of CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

Parameters	Description
Shape	The **Shape** parameter identifies the Shape object that will have another shape placed in front of it in the stacking order of CorelDRAW.

{button ,AL(`CLS_Shape;FNC_OrderIsInFrontOf')} **Related Topics**

Shape.Separate

Sub **Separate**()

Shape

Description

The **Separate** method separates any combined shape objects or a linked group in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_Shape;FNC_Separate')} **Related Topics**

Shape.Layer

Property **Layer** AS **Layer**

Shape

Description

The **Layer** property returns or sets a value associated with the properties of a layer upon which a shape object resides in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Example

The following code example displays the name of the layer containing the active shape in CorelDRAW:

```
Sub Test()
MsgBox ActiveShape.Layer.Name
End Sub
```

{button ,AL(`CLS_Shape;FNC_Layer')} **Related Topics**

Shape.Points

Property **Points** AS **Points**

Shape

Description

The **Points** property returns a value associated with the **Points** collection in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

This property returns a Read-Only value.

Example

The following code example displays the number of shape point objects in the Points collection of the active shape:

```
Sub PointsCollection()
With ActiveShape.Points
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Points')} **Related Topics**

Shape.Connector

Property **Connector** AS **Connector**

Shape

Description

The **Connector** property returns a value associated with the properties of a connector in CorelDRAW. A connector line object is a straight line that acts as a dynamic join between two objects in a drawing, created with either the Connector Line tool or the Interactive Connector tool. If you move one or both of the objects, the connector line (also called a flow line) is adjusted accordingly. If only one end of a line is connected to an object, the other end is fixed to the page. You can only reposition a connector line by moving the objects to which it is attached.

This property returns a Read-Only value.

Example

The following code example sets the reference point of the document to the lower left corner and creates a new shape point with coordinates (1, 6), one inch above and 6 inches to the right of the reference point. A rectangle and ellipse are created, and a connector line is drawn between them, starting at the tenth snap point of the rectangle to the first snap point of the ellipse:

```
Sub PointStart()
Dim sp as New ShapePoint
Dim s1 As Shape, s2 As Shape, c As Shape
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
Set s1 = ActiveLayer.CreateRectangle(0, 0, 3, 3)
Set s2 = ActiveLayer.CreateEllipse(4, 4, 6, 1)
Set c = ActiveLayer.CreateConnector(s1.Points(10), s2.Points(1))
    c.Connector.StartPoint = s1.Points(10)
    c.Connector.EndPoint = s2.Points(1)
End Sub
```

{button ,AL(`CLS_Shape;FNC_Connector')} **Related Topics**

Shape.IsOnShape

Function **IsOnShape**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal HotArea AS Double) AS **cdrPositionOfPointOverShape**

Shape

Description

The **IsOnCurve** method returns a value associated with a point (x and y coordinate) in relation to a shape object in CorelDRAW. A point may be inside, outside, or over the shape object. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The **IsOnCurve** method returns cdrPositionOfPointOverShape.

Parameters	Description
X	Sets the horizontal position (using the CorelDRAW rulers as a reference) for a point in the active document, created in relation to the reference point.
Y	Sets the vertical position (using the CorelDRAW rulers as a reference) for a point in the active document, created in relation to the reference point.
HotArea	Defines a restricted area on the shape in which to evaluate the position of a point, in relation to the shape. This value is optional.

{button ,AL(`CLS_Shape;FNC_IsOnShape')} **Related Topics**

Shape.CreateArrowHead

Function **CreateArrowHead**() AS **ArrowHead**

Shape

Description

The **CreateArrowhead** method creates an arrow from a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

{button ,AL(`CLS_Shape;FNC_CreateArrowHead')} **Related Topics**

Shape.Copy

Sub **Copy**()

Shape

Description

The **Copy** method creates a copy of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example copies the active shape in CorelDRAW:

```
Sub Test()
ActiveShape.Copy
End Sub
```

{button ,AL(`CLS_Shape;FNC_Copy')} **Related Topics**

Shape.Cut

Sub **Cut**()

Shape

Description

The **Cut** method cuts a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example cuts the active shape in CorelDRAW:

```
Sub Test()
ActiveShape.Cut
End Sub
```

{button ,AL(`CLS_Shape;FNC_Cut')} **Related Topics**

Shape.Clone

Function **Clone**(ByVal OffsetX AS Double, ByVal OffsetY AS Double) AS **Shape**

Shape

Description

The **Clone** method clones a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

Parameters	Description
OffsetX	The **OffsetX** parameter specifies the horizontal distance to offset the cloned shape object. This value is optional and the default value is 0.
OffsetY	The **OffsetY** parameter specifies the vertical distance to offset the cloned shape object. This value is optional and the default value is 0.

{button ,AL(`CLS_Shape;FNC_Clone')} **Related Topics**

Shape.Stretch

Sub **Stretch**(ByVal **StretchX** AS Double, ByVal StretchY AS Double, ByVal StretchCharactersSize AS Boolean)

Shape

Description

The **Stretch** method stretches a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stretching an object means to size an object horizontally or vertically. Stretching changes the size of an object in one direction only, as opposed to sizing, which maintains the aspect ratio (the ratio of height to width).

Parameters	Description
StretchX	The **StretchX** parameter specifies the distance in which to stretch the shape object horizontally.
StretchY	The **StretchY** parameter specifies the degree in which to stretch the shape object vertically.
StretchCharactersSize	Sets a True or False value that determines if characters are stretched with the shape. This value is optional and the default value is False.

{button ,AL(`CLS_Shape;FNC_Stretch')} **Related Topics**

Shape.SetPosition

Sub **SetPosition**(ByVal **PositionX** AS Double, ByVal **PositionY** AS Double)

Shape

Description

The **SetPosition** method sets the position of a shape object on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
PositionX	Sets the horizontal position of the shape object, in document units.
PositionY	Sets the vertical position of the shape object, in document units.

{button ,AL(`CLS_Shape;FNC_SetPosition')} **Related Topics**

Shape.SetSize

Sub **SetSize**(ByVal **Width** AS Double = 0, ByVal **Height** AS Double = 0)

Shape

Description

The **SetSize** method sets the size of a shape object on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Width	Sets the width of the shape object, in document units.
Height	Sets the height of the shape object, in document units.

{button ,AL(`CLS_Shape;FNC_SetSize')} **Related Topics**

Shape.GetPosition

Sub **GetPosition**(ByRef **PositionX** AS Double, ByRef **PositionY** AS Double)

Shape

Description

The **GetPosition** method gets the position of a shape object on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
PositionX	Sets the horizontal position of the shape object, in document units.
PositionY	Sets the vertical position of the shape object, in document units.

{button ,AL(`CLS_Shape;FNC_GetPosition')} **Related Topics**

Shape.GetSize

Sub **GetSize**(ByRef **Width** AS Double, ByRef **Height** AS Double)

Shape

Description

The **GetSize** method gets the size of a shape object on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Width	Sets the width of the shape object, in document units.
Height	Sets the height of the shape object, in document units.

{button ,AL(`CLS_Shape;FNC_GetSize')} **Related Topics**

Shape.Properties

Property **Properties** AS **Properties**

Shape

Description

The **Properties** property returns a value associated with the properties of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

This property returns a Read-Only value.

Example

The following code example counts the number of properties associated with the active shape:

```
Sub Test()
MsgBox ActiveShape.Properties.Count
End Sub
```

{button ,AL(`CLS_Shape;FNC_Properties')} **Related Topics**

Shape.OrderReverse

Sub **OrderReverse**()

Shape

Description

The **OrderReverse** method reverses the stacking order of shape objects in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_Shape;FNC_OrderReverse')} **Related Topics**

Shape.Combine

Function **Combine**() AS **Shape**

Shape

Description

The **Combine** method combines shape objects in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

When you combine rectangles, ellipses, polygons, stars, spirals, graphs, or text, CorelDRAW converts them to curves before converting them to a single curve object. When text is combined with other text, however, the text objects are not converted to curves; they are converted into larger blocks of text.

You can break apart any object that has been combined. If you break apart an object that has been created by combining Artistic text, the text breaks apart into lines first, then into words. Paragraph text, on the other hand, breaks into separate paragraphs. Both Artistic and Paragraph text can be recombined.

{button ,AL(`CLS_Shape;FNC_Combine')} **Related Topics**

Shape.BreakApart

Sub **BreakApart**()

Shape

Description

The **BreakApart** method breaks apart shape objects in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

You can break apart any object that has been combined. If you break apart an object that has been created by combining Artistic text, the text breaks apart into lines first, then into words. Paragraph text, on the other hand, breaks into separate paragraphs. Both Artistic and Paragraph text can be recombined.

{button ,AL(`CLS_Shape;FNC_BreakApart')} **Related Topics**

Shape.Weld

Function **Weld**(ByRef **TargetShape** AS **Shape**, ByVal LeaveSource AS Boolean, ByVal LeaveTarget AS Boolean) AS **Shape**

Shape

Description

The **Weld** method welds a shape object to another object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

You can create a single curve object out of two or more objects. If the objects overlap, the result is a single object with one outline. If the objects don't overlap, they form a weld group in which the objects appear separate but are actually treated as one object.

Parameters	Description
TargetShape	Specifies the object that welds with the shape object.
LeaveSource	Specifies a True or False value that determines whether or not to keep the shape object after the weld is complete. This value is optional and the default value is True.
LeaveTarget	Specifies a True or False value that determines whether or not to keep the target object after the weld is complete. This value is optional and the default value is True.

{button ,AL(`CLS_Shape;FNC_Weld')} **Related Topics**

Shape.Trim

Function **Trim**(ByRef **TargetShape** AS **Shape**, ByVal LeaveSource AS Boolean, ByVal LeaveTarget AS Boolean) AS **Shape**

Shape

Description

The **Trim** method trims a shape object to another object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Trimming creates new, irregularly shaped objects by removing the areas that overlap other selected objects. Before you trim objects, you must decide which object you want to trim (the target object) and which object(s) you want to use to trim it. The objects you use to trim must overlap - or be overlapped by - the target object. You can trim single objects with one or more objects, and you can trim multiple objects with multiple objects.

Parameters	Description
TargetShape	Specifies the object that trims with the shape object.
LeaveSource	Specifies a True or False value that determines whether or not to keep the shape object after the trim is complete. This value is optional and the default value is True.
LeaveTarget	Specifies a True or False value that determines whether or not to keep the target object after the trim is complete. This value is optional and the default value is True.

{button ,AL(`CLS_Shape;FNC_Trim')} **Related Topics**

Shape.Intersect

Function **Intersect**(ByRef **TargetShape** AS **Shape**, ByVal LeaveSource AS Boolean, ByVal LeaveTarget AS Boolean) AS **Shape**

Shape

Description

The **Intersect** method intersects a shape object with another object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Intersecting creates a new object from an area where two or more objects overlap. The result is one object with the size and shape of this overlapping area. In addition to this new object, you can keep all, some, or none of the original objects. The new object always uses the fill and outline attributes of the target object.

Parameters	Description
TargetShape	Specifies the object that intersects with the shape object.
LeaveSource	Specifies a True or False value that determines whether or not to keep the shape object after the intersection is complete. This value is optional and the default value is True.
LeaveTarget	Specifies a True or False value that determines whether or not to keep the target object after the intersection is complete. This value is optional and the default value is True.

{button ,AL(`CLS_Shape;FNC_Intersect')} **Related Topics**

Shape.Effects

Property **Effects** AS **Effects**

Shape

Description

The **Effects** property returns a value associated with the **Effects** collection in CorelDRAW.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

This property returns a Read-Only value.

Example

The following code example displays the effect type associated with the first effect in the Effects collection that is applied to the current shape in CorelDRAW. An effect must be applied to the active shape in order for the type to display in a message box:

```
Sub EffectsItem()
With ActiveShape.Effects
    MsgBox .Item(1).Type
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_Effects')} **Related Topics**

Shape.Effect

Property **Effect** AS **Effect**

Shape

Description

The **Effect** property returns a value associated with the effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

This property returns a Read-Only value.

Example

The following code example displays the effect type in the active shape in a message box:

```
Sub Test()
MsgBox ActiveShape.Effect.Type
End Sub
```

{button ,AL(`CLS_Shape;FNC_Effect')} **Related Topics**

Shape.CreateDropShadow

Function **CreateDropShadow**(ByVal Type AS **cdrDropShadowType**, ByVal Opacity AS Long, ByVal Feather AS Long, ByVal OffsetX AS Double, ByVal OffsetY AS Double, ByRef Color AS **Color**, ByVal FeatherType AS **cdrFeatherType**, ByVal FeatherEdge AS **cdrEdgeType**, ByVal PerspectiveAngle AS Double, ByVal PerspectiveStretch AS Double, ByVal Fade AS Long) AS **Effect**

Shape

Description

The **CreateDropShadow** method creates a drop shadow effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

Parameters	Description
Type	The **Type** property returns or sets the drop shadow type in a drop shadow effect in CorelDRAW. This property returns cdrDropShadowType. This value is optional and the default value is **cdrDropShadowFlat** (0)
Opacity	The **Opacity** property returns the opacity value of a drop shadow effect in CorelDRAW. Opacity determines the intensity of a drop shadow effect. You can type values between 0 and 100. Low values create a less opaque drop shadow, while high values create a more opaque drop shadow. This value is optional and the default value is 50.
Feather	The **Feather** property returns the feather length for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect. This value is optional and the default value is **15**
OffsetX	The **OffsetX** property returns the vertical offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object. This value is optional and the default value is 0.
OffsetY	The **OffsetY** property returns the vertical offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object. This value is optional and the default value is 0.
Color	The **Color** property returns or sets the color of a drop shadow effect in CorelDRAW. This value is optional and the default value is **Nothing** (0)
FeatherType	The **FeatherType** property returns the feather type for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect. This property returns cdrFeatherType.This value is optional and the default value is **cdrFeatherAverage** (3)
FeatherEdge	The **FeatherEdge** property returns the feather edge for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect. This property returns cdrEdgeType. This value is optional and the default value is cdrEdgeLinear (0)
PerspectiveAngle	The **PersectiveAngle** returns or set the feather perspective angle in a drop shadow effect in CorelDRAW. This value is optional and the default value is -45.

PerspectiveStretch	The **PerspectiveStretch** returns or set the feather perspective stretch value in a drop shadow effect in CorelDRAW. This value is optional and the default value is 1.
Fade	The **Fade** property returns or set the fade value of a drop shadow effect in CorelDRAW. You can't change the fade level of a drop shadow which has a Flat perspective. This value is optional and the default value is 1.

{button ,AL(`CLS_Shape;FNC_CreateDropShadow')} **Related Topics**

Shape.CreateBlend

Function **CreateBlend**(ByRef **Shape** AS **Shape**, ByVal Steps AS Integer, ByVal ColorBlendType AS **cdrFountainFillBlendType**, ByVal Mode AS **cdrBlendMode**, ByVal Spacing AS Integer, ByVal Angle AS Double, ByVal Loop AS Boolean, ByRef Path AS **Shape**, ByVal RotateShapes AS Boolean, ByVal SpacingAccel AS Long, ByVal ColorAccel AS Long, ByVal AccelSize AS Boolean) AS **Effect**

Shape

Description

The **CreateBlend** method creates a blend effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

Parameters	Description
Shape	The **Shape** property returns or set the starting shape in a blend effect in CorelDRAW.
Steps	The **Steps** property returns or sets the number of steps in a blend effect in CorelDRAW. The number of steps in a contour effect is linked to the number of lines that appear in the effect. This value is optional and the default value is 20.
ColorBlendType	The **ColorBlendType** property returns or set the color blend type of a fountain fill blend type in a blend effect in CorelDRAW. This property returns cdrFountainFillBlendType. This value is optional and the default value is **cdrDirectFountainFillBlend** (0)
Mode	The **Mode** property returns or set the blend mode of a blend effect in CorelDRAW. This property returns cdrBlendMode. This value is optional and the default value is **cdrBlendSteps** (0)
Spacing	The **Spacing** property returns or sets the blend spacing in a blend effect in CorelDRAW. This value is optional and the default value is 1.
Angle	The **Angle** property returns or sets the angle in a blend effect in CorelDRAW. This value is optional and the default value is 0.
Loop	The **Loop** property returns or sets a True or False value that indicates whether or not a blend loops in a blend effect in CorelDRAW. Looping allows you to rotate the blend halfway between the start and end objects' centers of rotation. This value is optional and the default value is False.
Path	The **Path** property returns or sets the blend path in a blend effect in CorelDRAW. This value is optional and the default value is Nothing (0).
RotateShapes	The **RotateShapes** property returns or sets a True or False value that indicates whether or not the blend effect rotates its shape to follow a path in CorelDRAW. This value is optional and the default value is False.
SpacingAccel	The **SpacingAccel** property returns or sets the rate of object acceleration in a blend effect in CorelDRAW. You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate. This value is optional and the default value is 0.
ColorAccel	The **ColorAccel** property returns or sets the rate of color acceleration in a blend effect in CorelDRAW. Higher numbers allow the colors to move quicker through the spectrum as they approach the end object. This value is optional and the default value is 0.
AccelSize	The **AccelSize** property returns or sets a True or False value that indicates whether or not the size of a blend accelerates along the path in a blend effect in CorelDRAW. This value is optional and the

default value is False.

{button ,AL(`CLS_Shape;FNC_CreateBlend')} **Related Topics**

Shape.CreateExtrude

Function **CreateExtrude**(ByVal **Type** AS **cdrExtrudeType**, ByVal **VPType** AS **cdrExtrudeVPType**, ByVal **VPX** AS Double, ByVal **VPY** AS Double, ByVal Depth AS Double, ByVal Shading AS **cdrExtrudeShading**, ByRef BaseColor AS **Color**, ByRef ShadingColor AS **Color**, ByVal BevelDepth AS Double, ByVal BevelAngle AS Double, ByRef BevelColor AS **Color**, ByVal BevelOnly AS Boolean) AS **Effect**

Shape

Description

The **CreateExtrude** method creates an extrude effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
Type	The **Type** property returns or sets the extrude type in an extrude effect in CorelDRAW. This property returns cdrExtrudeType.
VPType	The **VPType** property returns or set the vanishing point in an extrude effect in CorelDRAW.
VPX	Sets the X coordinate of the vanishing point in the extrude effect.
VPY	Sets the X coordinate of the vanishing point in the extrude effect.
Depth	The **Depth** property returns or sets the extrude depth in an extrude effect in CorelDRAW. This value is optional and the default value is 20.
Shading	The **Shading** property returns or sets the shading type in an extrude effect in CorelDRAW. This property returns cdrExtrudeShading. This value is optional and the default value is **cdrExtrudeObjectFill** (0)
BaseColor	The **BaseColor** property returns or sets the solid fill color or starting shading color in an extrude effect in CorelDRAW.This value is optional and the default value is Nothing (0).
ShadingColor	The **ShadingColor** property returns or sets the ending shading color in an extrude effect in CorelDRAW. This value is optional and the default value is Nothing (0).
BevelDepth	The **BevelDepth** property returns or sets the bevel depth in an extrude effect in CorelDRAW. This value is optional and the default value is 0.
BevelAngle	The **BevelAngle** returns or sets the bevel angle in an extrude effect in CorelDRAW. This value is optional and the default value is 45.
BevelColor	The **BevelColor** property returns or sets the bevel color in an extrude effect in CorelDRAW. This value is optional and the default value is Nothing (0).
BevelOnly	The **BevelOnly** property returns or sets a True or False value that indicates whether or not to use only a bevel in an extrude effect in CorelDRAW. This value is optional and the default value is False.

{button ,AL(`CLS_Shape;FNC_CreateExtrude')} **Related Topics**

Shape.CreateEnvelope

Function **CreateEnvelope**(ByVal **PresetIndex** AS Long, ByVal Mode AS **cdrEnvelopeMode**, ByVal KeepLines AS Boolean) AS **Effect**

Shape

Description

The **CreateEnvelope** method creates an envelope effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

Parameters	Description
PresetIndex	The **PresetIndex** parameter is an index number that uniquely identifies a preset envelope effect in CorelDRAW.
Mode	The **Mode** property returns or sets the mapping mode type for the envelope effect in CorelDRAW. The **Mode** property returns a value of cdrEnvelopeMode. This value is optional and the default value is **cdrEnvelopePutty** (2)
KeepLines	The **KeepLines** property returns or sets a True or False value that indicates if an envelope effect keeps its lines straight or converts them to curves in CorelDRAW. If the **KeepLines** property is True, the envelope effect keeps its lines straight - it does not convert the lines to curves. This value is optional and the default value is False.

{button ,AL(`CLS_Shape;FNC_CreateEnvelope')} **Related Topics**

Shape.Flip

Sub **Flip**(ByVal **Axes** AS **cdrFlipAxes**)

Shape

Description

The **Flip** method flips a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A shape can be flipped horizontally, vertically, or both.

Parameters	Description
Axes	Sets the manner of the flip in the shape object. This value returns cdrFlipAxes.

Example

The following code example flips the active shape horizontally in CorelDRAW:

```
Sub Test()
ActiveShape.Flip cdrFlipHorizontal
End Sub
```

{button ,AL(`CLS_Shape;FNC_Flip')} **Related Topics**

cdrFlipHorizontal=1

cdrFlipVertical=2

cdrFlipBoth=3

Shape.Locked

Property **Locked** AS Boolean

Shape

Description

The **Locked** property returns or sets a True or False value that indicates whether or not a shape object is locked in CorelDRAW. If lock, not edits can be made to the object. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example displays the locked status of the active shape:

```
Sub Lock()
MsgBox ActiveShape.Locked
End Sub
```

{button ,AL(`CLS_Shape;FNC_Locked')} **Related Topics**

Shape.OriginalWidth

Property **OriginalWidth** AS Double

Shape

Description

The **OriginalWidth** property returns the original width size of a shape object when it was created in CorelDRAW, before any adjustments have been made to the object. A shape is an object that can be displayed as several variations of a rectangle or a circle.

This property returns a Read-Only value.

Example

The following code example displays the original width of the active shape:

```
Sub Test()
MsgBox ActiveShape.OriginalWidth
End Sub
```

{button ,AL(`CLS_Shape;FNC_OriginalWidth')} **Related Topics**

Shape.OriginalHeight

Property **OriginalHeight** AS Double

Shape

Description

The **OriginalHeight** property returns the original height size of a shape object when it was created in CorelDRAW, before any adjustments have been made to the object. A shape is an object that can be displayed as several variations of a rectangle or a circle.

This propert returns a Read-Only value.

Example

The following code example displays the original height of the active shape:

```
Sub Test()
MsgBox ActiveShape.OriginalHeight
End Sub
```

{button ,AL(`CLS_Shape;FNC_OriginalHeight')} **Related Topics**

Shape.Selected

Property **Selected** AS Boolean

Shape

Description

The **Selected** property returns or sets a True or False value that indicates whether a shape object is selected in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example sets the active shape's **Selected** property to True:

```
Sub Test()
ActiveShape.Selected = True
End Sub
```

{button ,AL(`CLS_Shape;FNC_Selected')} **Related Topics**

Shape.CreateLens

Function **CreateLens**(ByVal **Type** AS **cdrLensType**, ByVal RateOrMagnification AS Double, ByRef Color1 AS **Color**, ByRef Color2 AS **Color**, ByVal ColorMapPalette AS **cdrFountainFillBlendType**) AS **Effect**

Shape

Description

The **CreateLens** method creates a lens effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

When you apply a lens to an object, you change the way you perceive the objects behind it. You can choose amongst several types of lenses, each producing distinctive results. These results range from color alteration (as produced by heat map, inverting, and brightening lenses, for example) to distortion (as produced by magnifying and fish eye lenses). In each case, the lens changes the way you perceive the objects behind it, not the actual properties and attributes of those objects.

Parameters	Description
Type	The **Type** property returns or sets the lens type in a lens effect in CorelDRAW. This property returns cdrLensType.
RateOrMagnification	The **Rate** property applies to various lens effects, including Brighten, Color Add, Color Limit, and Fish eye lens effects. Rate refers to values between 0 and 100% that increase degree of a lens effect. This value is optional and the default value is 50.
Color1	The **Color1** property returns or sets the starting color in a Custom Color Map Lens effect in CorelDRAW. This value is optional and the default value is Nothing (0).
Color2	The **Color2** property returns or sets the ending color in a Custom Color Map Lens effect in CorelDRAW. This value is optional and the default value is Nothing (0).
ColorMapPalette	The **ColorMapPalette** property returns or sets the color map palette type in a lens effect in CorelDRAW. This value returns cdrFountainFillBlendType. This value is optional and the default value is **cdrDirectFountainFillBlend** (0)

{button ,AL(`CLS_Shape;FNC_CreateLens')} **Related Topics**

Shape.CreatePerspective

Function **CreatePerspective**(ByVal HorizVanishPointX AS Variant, ByVal HorizVanishPointY AS Variant, ByVal VertVanishPointX AS Variant, ByVal VertVanishPointY AS Variant) AS **Effect**

Shape

Description

The **CreatePerspective** method creates a perspective effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

Parameters	Description
HorizVanishPointX	The **HorizVanishPointX** property returns or sets the horizontal vanishing point used in a perspective effect in CorelDRAW. You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate. This value is optional.
HorizVanishPointY	The **HorizVanishPointY** property returns or sets the horizontal vanishing point used in a perspective effect in CorelDRAW. You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate. This value is optional.
VertVanishPointX	The **VertVanishPointX** property returns or sets the vertical vanishing point used in a perspective effect in CorelDRAW. You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate. This value is optional.
VertVanishPointY	The **VertVanishPointY** property returns or sets the vertical vanishing point used in a perspective effect in CorelDRAW. You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate. This value is optional.

{button ,AL(`CLS_Shape;FNC_CreatePerspective')} **Related Topics**

Shape.CreateContour

Function **CreateContour**([ByVal Direction As **cdrContourDirection** = cdrContourOutside (1)], [ByVal Offset As Double = 0.150000005960464], [ByVal Steps As Long = 1], [ByVal BlendType As **cdrFountainFillBlendType** = cdrDirectFountainFillBlend (0)], [ByVal OutlineColor As **Color** = 0], [ByVal FillColor As **Color** = 0], [ByVal FillColor2 As **Color** = 0], [ByVal SpacingAccel As Long = 0], [ByVal ColorAccel As Long = 0]) As **Effect**

Shape

Description

The **CreateContour** method creates a contour effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

The spaces between contour lines are filled with colors that follow a progression from the original object to the last shape created. If there is a difference in color between the contour lines and the outline of the original object, a second progression occurs. You can modify both color progressions to get the look you want.

Parameters	Description
Direction	The **Direction** property returns or sets the direction of a contour effect in CorelDRAW. This property returns a value of cdrContourDirection. This value is optional and the default value is **drContourOutside** (1)
Offset	The **Offset** property returns or sets the offset distance, in document units, between contour lines in a contour effect in CorelDRAW. This automatically adjusts the number of contour steps. This value is optional and the default value is .1.
Steps	The **Steps** property returns or sets the number of steps in a contour effect in CorelDRAW. The number of steps in a contour effect is linked to the number of lines that appear in the effect. This value is optional and the default value is 1.
BlendType	The **BlendType** property returns or set the color blend type of a fountain fill blend type in a contour effect in CorelDRAW. This property returns cdrFountainFillBlendType. This value is optional and the default value is **cdrDirectFountainFillBlend** (0)
OutlineColor	The **OutlineColor** property returns or sets the contour outline color in CorelDRAW. This value is optional and the default value is Nothing (0).
FillColor	The **FillColor** property returns or sets the contour fill color in CorelDRAW. This value is optional and the default value is Nothing (0).
FillColor2	The **FillColor2** property returns or sets the FillTo color of a contour effect in CorelDRAW. This value is optional and the default value is Nothing (0).

{button ,AL(`CLS_Shape;FNC_CreateContour')} **Related Topics**

Shape.CreatePushPullDistortion

Function **CreatePushPullDistortion**(ByVal **OriginX** AS Double, ByVal **OriginY** AS Double, ByVal **Amplitude** AS Long) AS **Effect**

Shape

Description

The **CreatePushPullDistortion** method creates a push pull distortion effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

When you apply a Push distortion to an object, the object's nodes are forced towards its center thereby rounding the object. The Pull distortion draws an object's nodes away from its center, making the object pointy.

When you create a Push or Pull distortion you like, you can copy it and apply it to other selected objects. If you don't like the distortion, you can remove it without deleting the object.

Parameters	Description
OriginX	Sets the x coordinate in the shape object, around which the push pull distortion rotates.
OriginY	Sets the y coordinate in the shape object, around which the push pull distortion rotates.
Amplitude	**Amplitude** determines the intensity of a push pull distortion effect. You can also adjust the amplitude of the effect by typing a value in the Push Pull Distortion Amplitude box on the Property Bar. You can type values from 0 to 100. Higher values produce a more pronounced distortion.

{button ,AL(`CLS_Shape;FNC_CreatePushPullDistortion')} **Related Topics**

Shape.CreateZipperDistortion

Function **CreateZipperDistortion**(ByVal **OriginX** AS Double, ByVal **OriginY** AS Double, ByVal **Amplitude** AS Long, ByVal **Frequency** AS Long, ByVal Random AS Boolean, ByVal Smooth AS Boolean, ByVal Local AS Boolean) AS **Effect**

Shape

Description

The **CreateZipperDistortion** method creates a zipper distortion effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

Parameters	Description
OriginX	Sets the x coordinate in the shape object, around which the zipper distortion rotates.
OriginY	Sets the y coordinate in the shape object, around which the zipper distortion rotates.
Amplitude	**Amplitude** determines the intensity of a zipper distortion effect. You can also adjust the amplitude of the zipper effect by typing a value in the Zipper Distortion Amplitude box on the Property Bar. You can type values from 0 to 100. Higher values produce a more pronounced zipper distortion.
Frequency	The **Frequency** property can have a range of values from 0 to 100. A higher frequency value creates more zipper points in the distortion effect.
Random	The **Random** property returns or sets a True or False value that indicates whether the zipper distortion effect is a random effect. With a random Zipper distortion, the horizontal and vertical points follow a haphazard course. This value is optional and the default value is False.
Smooth	The **Smooth** property returns or sets a True or False value that indicates whether the zipper distortion effect is a smooth effect. A smooth distortion lets you smoothen the points of the Zipper distortion, instead of having them appear as jagged edges. This value is optional and the default value is False.
Local	The **Local** property returns or sets a True or False value that indicates whether the zipper distortion effect is a local effect. A local distortion lets you emphasize the Zipper distortion in a specific area of the selected object. This value is optional and the default value is False.

{button ,AL(`CLS_Shape;FNC_CreateZipperDistortion')} **Related Topics**

Shape.CreateTwisterDistortion

Function **CreateTwisterDistortion**(ByVal **OriginX** AS Double, ByVal **OriginY** AS Double, ByVal **Angle** AS Double) AS **Effect**

Shape

Description

The **CreateTwisterDistortion** method creates a twister distortion effect in a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

The Twister distortion turns an object around itself. When you apply a Twister distortion to an object, the point at which you click becomes the center of the distortion, which can be also identified by a diamond-shaped handle. The center is fixed while one end of the object is wound around this fixed point. A dashed line, the Horizontal Line Of Origin, extends from the center of the object. This line measures the amount of Twister distortion in degrees you apply as you drag the rotation handle in either a clockwise or counterclockwise direction. One full rotation is 359 degrees.

Parameters	Description
OriginX	Sets the x coordinate in the shape object, around which the twister distortion rotates.
OriginY	Sets the y coordinate in the shape object, around which the twister distortion rotates.
Angle	Sets the degree of the twister rotation effect in the shape object.

{button ,AL(`CLS_Shape;FNC_CreateTwisterDistortion')} **Related Topics**

Shape.Guide

Property **Guide** AS **Guide**

Shape

Description

The **Guide** property returns a value associated with a guide object in CorelDRAW.A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guidelines may be Horizontal, Vertical, or Slanted and any number of them may be created and saved with your drawing. Guidelines return a value of cdrGuideType. You can also enable snapping guidelines so that objects automatically align with the guidelines when moved or drawn nearby.

This property returns a Read-Only value.

Example

The following code example looks at all the guidelines in the master document and uses the MakeEditable method to convert all preset guidelines in the active document to standard guidelines:

```
Sub EditGuideline()
Dim MyShape As Shape
For Each MyShape In ActiveDocument.Pages(0).Guides(cdrAllGuides)
'looks at each guideline in the master document
'0 denotes the master page
    If MyShape.Guide.Preset Then
    MyShape.Guide.MakeEditable
    End If
Next MyShape
End Sub
```

{button ,AL(`CLS_Shape;FNC_Guide')} **Related Topics**

Shape.AddToPowerClip

Sub **AddToPowerClip**(ByRef **Shape** AS **Shape**)

Shape

Description

The **AddToPowerClip** method adds a shape object to a powerclip object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

Parameters	Description
Shape	The **Shape** parameter identifies the shape object that is added to the powerclip in CorelDRAW.

{button ,AL(`CLS_Shape;FNC_AddToPowerClip')} **Related Topics**

Shape.PowerClip

Property **PowerClip** AS **PowerClip**

Shape

Description

The **PowerClip** property returns a value associated with the powerclip object of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

This property returns a Read-Only value.

Example

The following code example unlocks the contents of the active shape's powerclip object, counts the number of shapes contained in the powerclip, and extracts the shape objects. The code leaves edit mode when the shapes have been extracted, and the contents of the powerclip are locked:

```
Sub PowerClipEdit()
With ActiveShape.PowerClip
    .ContentsLocked = False
    .EnterEditMode
    .ExtractShapes
    .LeaveEditMode
    .ContentsLocked = True
End With
End Sub
```

{button ,AL(`CLS_Shape;FNC_PowerClip')} **Related Topics**

Shape.PowerClipParent

Property **PowerClipParent** AS **Shape**

Shape

Description

The **PowerClipParent** property returns a value associated with the powerclip parent object of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

This property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Shape;FNC_PowerClipParent')} **Related Topics**

Shape.DrapeFill

Property **DrapeFill** AS Boolean

Shape

Description

The **DrapeFill** property returns a True or False vale that indicates whether or not to drape the fill of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can drape the control object's fill so that it covers the entire extrusion seamlessly.

Example

The following code example sets the **DrapeFill** property to True in the active shape:

```
Sub Outline()
ActiveShape.DrapeFill = True
End Sub
```

{button ,AL(`CLS_Shape;FNC_DrapeFill')} **Related Topics**

Shape.OverprintFill

Property **OverprintFill** AS Boolean

Shape

Description

The **OverprintFill** property returns a True or False vale that indicates whether or not to overprint the fill of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

Example

The following code example sets the **OverprintFill** property to True in the active shape:

```
Sub Outline()
ActiveShape.OverprintFill = True
End Sub
```

{button ,AL(`CLS_Shape;FNC_OverprintFill')} **Related Topics**

Shape.OverprintOutline

Property **OverprintOutline** AS Boolean

Shape

Description

The **OverprintOutline** property returns a True or False vale that indicates whether or not to overprint the outline of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

Example

The following code example sets the **OverprintOutline** property to True in the active shape:

```
Sub Outline()
ActiveShape.OverprintOutline = True
End Sub
```

{button ,AL(`CLS_Shape;FNC_OverprintOutline')} **Related Topics**

Shape.URL

Property **URL** As **URL**

Gets the URL Object

Member of **Shape**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Shape;FNC_URL')} **Related Topics**

Shape.ObjectData

Property **ObjectData** AS **DataItems**

Shape

Description

The **ObjectData** property returns the object data value of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Object data contains information about each object in CorelDRAW. Its information is accessed through the Object Manager.

This property returns a Read-Only value.

{button ,AL(`CLS_Shape;FNC_ObjectData')} **Related Topics**

Shape.CloneLink

Property **CloneLink** AS **CloneLink**

Shape

Description

The **CloneLink** property returns the properties of a clone link object in CorelDRAW. _ENG Gets the clone link properties

A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

The CloneLink class represents the properties of the actual link between clone and parent objects. Objects are linked in Object Linking and Embedding (OLE) when information from one file (the source file) is inserted in another file (the destination file). The source file is then linked to the destination file. Changes made to the information in the source file can be automatically or manually updated in the destination file.

This property returns a Read-Only value.

Example

The following code example selects a clone object in the active document and displays the name of its parent object in a message box:

```
Sub ParentClone()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
'note that the selected object must be the clone object
MsgBox s.CloneLink.CloneParent.Name
End Sub
```

{button ,AL(`CLS_Shape;FNC_CloneLink')} **Related Topics**

Shape.Clones

Property **Clones** AS **ShapeRange**

Shape

Description

The **Clones** property returns a value associated with a collection of clone objects in CorelDRAW. A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

You can also clone a special effect that is applied to an object and apply it to other objects. Objects with a cloned effect take on all changes that are made to that effect in the master.

This property returns a Read-Only value.

{button ,AL(`CLS_Shape;FNC_Clones')} **Related Topics**

Shape.AbsoluteHScale

Property **AbsoluteHScale** AS Double

Shape

Description

The **AbsoluteHScale** property returns a value associated with the absolute horizontal scale of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Scaling allows you to change an object's horizontal and vertical dimensions or to maintain the aspect ratio. Scaling alters the object's dimensions by a specified percentage.

This property returns a Read-Only value.

Example

The following code example displays the absolute horizontal scale value in a message box:

```
Sub Test()
MsgBox ActiveShape.AbsoluteHScale
End Sub
```

{button ,AL(`CLS_Shape;FNC_AbsoluteHScale')} **Related Topics**

Shape.AbsoluteVScale

Property **AbsoluteVScale** AS Double

Shape

Description

The **AbsoluteVScale** property returns a value associated with the absolute vertical scale of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Scaling allows you to change an object's horizontal and vertical dimensions or to maintain the aspect ratio. Scaling alters the object's dimensions by a specified percentage.

This property returns a Read-Only value.

Example

The following code example displays the absolute vertical scale value in a message box:

```
Sub Test()
MsgBox ActiveShape.AbsoluteVScale
End Sub
```

{button ,AL(`CLS_Shape;FNC_AbsoluteVScale')} **Related Topics**

Shape.AbsoluteSkew

Property **AbsoluteSkew** AS Double

Shape

Description

The **AbsoluteSkew** property returns a value associated with the absolute skew of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

This property returns a Read-Only value.

Example

The following code example displays the absolute skew value in a message box:

```
Sub Test()
MsgBox ActiveShape.AbsoluteSkew
End Sub
```

{button ,AL(`CLS_Shape;FNC_AbsoluteSkew')} **Related Topics**

Shape.Transparency

Property **Transparency** AS **Transparency**

Shape

Description

The **Transparency** property returns the transparency value of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

This property returns a Read-Only value.

Example

The following code example displays a value associated with the transparency type in the active shape of CorelDRAW:

```
Sub Trans()
MsgBox ActiveShape.Transparency.Type
End Sub
```

{button ,AL(`CLS_Shape;FNC_Transparency')} **Related Topics**

Shape.GetMatrix

Sub **GetMatrix**(ByRef **d11** AS Double, ByRef **d12** AS Double, ByRef **d21** AS Double, ByRef **d22** AS Double, ByRef **tx** AS Double, ByRef **ty** AS Double)

Shape

Description

The **GetMatrix** method gets the matrix for a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A matrix acts as a mould for a shape object in CorelDRAW.

Parameters	Description
d11	Sets a boundary for the matrix.
d12	Sets a boundary for the matrix.
d21	Sets a boundary for the matrix.
d22	Sets a boundary for the matrix.
tx	Sets the x coordinate for the center of the matrix.
ty	Sets the x coordinate for the center of the matrix.

{button ,AL(`CLS_Shape;FNC_GetMatrix')} **Related Topics**

Shape.SetMatrix

Sub **SetMatrix**(ByVal **d11** AS Double, ByVal **d12** AS Double, ByVal **d21** AS Double, ByVal **d22** AS Double, ByVal **tx** AS Double, ByVal **ty** AS Double)

Shape

Description

The **SetMatrix** method sets the matrix for a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A matrix acts as a mould for a shape object in CorelDRAW.

Parameters	Description
d11	Sets a boundary for the matrix.
d12	Sets a boundary for the matrix.
d21	Sets a boundary for the matrix.
d22	Sets a boundary for the matrix.
tx	Sets the x coordinate for the center of the matrix.
ty	Sets the x coordinate for the center of the matrix.

{button ,AL(`CLS_Shape;FNC_SetMatrix')} **Related Topics**

Shape.ConvertToBitmapEx

Function **ConvertToBitmapEx**(ByVal Mode AS **cdrImageType**, ByVal Dithered AS Boolean, ByVal Transparent AS Boolean, ByVal Resolution AS Long, ByVal AntiAliasing AS **cdrAntiAliasingType**, ByVal UseColorProfile AS Boolean) AS **Shape**

Shape

Description

The **ConvertToBitmapEx** converts a shape object in to a bitmap image in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Parameters	Description
Mode	Sets the image type, and returns cdrImageType. This value is optional and the default value is cdrRGBColorImage (4).
Dithered	Set to TRUE (-1) enables dithering. This value is optional and the default value is False.
Transparent	Set to TRUE (-1) enables transparent background. This value is optional and the default value is False.
Resolution	Lets you specify the resolution. This value is optional and the default value is 72.
AntiAliasing	Lets you specify the anti aliasing, which returns cdrAntiAliasingType. This value is optional and the default value is cdrNormalAntiAliasing (1).
UseColorProfile	Set to TRUE (-1) uses the color profile. This value is optional and the default value is True.

{button ,AL(`CLS_Shape;FNC_ConvertToBitmapEx')} **Related Topics**

Shape.SkewEx

Sub **SkewEx**(ByVal **AngleX** AS Double, ByVal **AngleY** AS Double, ByVal **CenterX** AS Double, ByVal **CenterY** AS Double)

Shape

Description

The **SkewEx** method skews a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

Parameters	Description
AngleX	Sets the horizontal angle, in degrees, in which the shape object is skewed.
AngleY	Sets the vertical angle, in degrees, in which the shape object is skewed.
CenterX	Sets the x coordinatefor the center of rotation in the shape object, around which the object is skewed.
CenterY	Sets the Y coordinate for the center of rotation in the shape object, around which the object is skewed.

{button ,AL(`CLS_Shape;FNC_SkewEx')} **Related Topics**

Shape.RotateEx

Sub **RotateEx**(ByVal **Angle** AS Double, ByVal **CenterX** AS Double, ByVal **CenterY** AS Double)

Shape

Description

The **RotateEx** method rotates a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Rotating an object means to reposition and reorient the object by turning it around its center of rotation.

Parameters	Description
Angle	Sets the angle measurement, in degrees, in which the shape object is rotated.
CenterX	Sets the x coordinatefor the center of rotation in the shape object, around which the object rotates.
CenterY	Sets the Y coordinate for the center of rotation in the shape object, around which the object rotates.

{button ,AL(`CLS_Shape;FNC_RotateEx')} **Related Topics**

Shape.ParentGroup

Property **ParentGroup** AS **Shape**

Shape

Description

The **ParentGroup** property returns a value associated with the parent group of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A parent group is a grouping of shape objects that contain other groupings, called child groups, within the parent group.

This property returns a Read-Only value.

{button ,AL(`CLS_Shape;FNC_ParentGroup')} **Related Topics**

Shape.SetBoundingBox

Sub **SetBoundingBox**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal **Width** AS Double, ByVal **Height** AS Double, ByVal KeepAspect AS Boolean, ByVal ReferencePoint AS **cdrReferencePoint**)

Shape

Description

The **SetBoundingBox** method sets the bounding box around a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The bounding box is the area that surrounds a shape object in CorelDRAW. The bounding box represents the maximum space a shape object can occupy in CorelDRAW. You can resize the bounding box.

Parameters	Description
X	Sets the horizontal position, x coordinate, of the bounding box's center, in document units.
Y	Sets the vertical position, y coordinate, of the bounding box's center, in document units.
Width	Sets the width of the bounding box, in document units..
Height	Sets the height of the bounding box, in document units..
KeepAspect	Keeps the size proportions of the bounding box in relation to its contents. The value is optional and the default value is False.
ReferencePoint	Sets the reference point in CorelDRAW. This value is optional and the default value is cdrCenter (9).

{button ,AL(`CLS_Shape;FNC_SetBoundingBox')} **Related Topics**

GMSManager properties

GMSManager Legend

- GMSPath

GMSManager methods

GMSManager Legend

RunMacro

GMSManager

Class **GMSManager**

Properties Methods Referenced By

GMSManager Class

{button ,AL(`CLS_GMSManager')} **Related Topics**

GMSManager.GMSPath

Property **GMSPath** As String

Gets a path to the folder where all GMS modules are stored

Member of **GMSManager**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GMSManager;FNC_GMSPath')} **Related Topics**

GMSManager.RunMacro

Function **RunMacro**(ByVal **ModuleName** As String, ByVal **MacroName** As String, ByRef **Parameters**() As Variant) As Variant

Runs a macro in a GMS module

Member of **GMSManager**

Parameters	Description
ModuleName	Description of ModuleName goes here (**in**)
MacroName	Description of MacroName goes here (**in**)
Parameters	Description of Parameters goes here

Return value

Returns Variant

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_GMSManager;FNC_RunMacro')} **Related Topics**

FrameWork properties

FrameWork Legend

- CommandBars
- MainMenu
-
- Name
- StatusBar

FrameWork methods

FrameWork Legend

ImportWorkspace

FrameWork.ImportWorkspace

Sub **ImportWorkspace**(ByVal **FileName** As String)

Imports a workspace into the framework

Member of **FrameWork**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_FrameWork;FNC_ImportWorkspace')} **Related Topics**

FrameWork

Class **FrameWork**

Properties Referenced by

The **FrameWork** class defines the characteristics of CorelDRAW tool objects and describes the look and behavior of the objects through their properties and methods. The FrameWork object in CorelDRAW acts as a container for the tools that you use to construct and view your CorelDRAW objects.

There are several features contained within the framework of CorelDRAW. The Command Bars, Menus, Status Bar, Toolbars, Main Menu, and Key Tables are primary components of any CorelDRAW document.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar, or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A key table is a file that contains lists of shortcut keys. Shortcut keys are used to speed up, or "accelerate," editing tasks. Different tables are active depending on what you're doing. For example, when you highlight text the Text Editing accelerator table becomes active. If no text is selected, the Main accelerator table is active.

{button ,AL(`CLS_FrameWork')} **Related Topics**

FrameWork.CommandBars

Property **CommandBars** AS **CommandBars**

FrameWork

Description

The **CommandBars** property returns a value associated with command bars in the **CommandBars** collection of CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar, or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **CommandBars** property returns a Read-Only value.

Example

The following code example displays the number of command bars in the Framework of CorelDRAW:

```
Sub FrameCommand()
With FrameWork
    MsgBox .CommandBars.Count
End With
End Sub
```

{button ,AL(`CLS_FrameWork;FNC_CommandBars')} **Related Topics**

FrameWork.MainMenu

Property **MainMenu** AS **CommandBar**

FrameWork

Description

The **MainMenu** property returns a value associated with the Main Menu in CorelDRAW. A menu is a list of commands that appears when you click a menu name in the Menu Bar.

The Main Menu in CorelDRAW is located above the Main Toolbar and Property bar, at the top of the Main Application Window. The Main Menu contains the most popular and standardized commands within the CorelDRAW application and include the File, Edit, View, Layout, Arrange, Effects, Bitmaps, Text, Tools, Window and Help menus.

The **MainMenu** property returns a Read-Only value.

Example

The following code example counts the number of control located in the Main Menu of CorelDRAW:

```
Sub MenuHide()
With FrameWork
    .MainMenu.Enabled = False
End With
End Sub
```

{button ,AL(`CLS_FrameWork;FNC_MainMenu')} **Related Topics**

FrameWork.StatusBar

Property **StatusBar** AS **CommandBar**

FrameWork

Description

The **StatusBar** property returns a value associated with the status bar in CorelDRAW. The Status Bar is an on-screen display area that shows information about objects, ongoing operations, and mouse position. You can specify the contents, appearance, and location of the Status Bar in the Application Window.

The **StatusBar** property returns a Read-Only value.

Example

The following code example displays the number of status bar controls in the Framework of CorelDRAW:

```
Sub FrameCommand()
With FrameWork
    MsgBox .StatusBar.Controls.Count
End With
End Sub
```

{button ,AL(`CLS_FrameWork;FNC_StatusBar')} **Related Topics**

FrameWork.Name

Property **Name** AS String

FrameWork

Description

The **Name** property returns a string value that uniquely identifies the FrameWork object in CorelDRAW.The FrameWork object in CorelDRAW acts as a container for the tools that you use to construct and view your CorelDRAW objects.

There are several features contained within the framework of CorelDRAW. The Command Bars, Menus, Status Bar, Toolbars, Main Menu, and Key Tables are primary components of any CorelDRAW document.

The **Name** property returns a Read-Only value.

Example

The following code example displays the name of the framework object in a message box:

```
Sub Frame()
With FrameWork
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_FrameWork;FNC_Name')} **Related Topics**

CommandBars properties

CommandBars Legend

- Count

- Item

CommandBars methods

CommandBars Legend

Add

CommandBars

Class **CommandBars**

Properties Methods Referenced by

The **CommandBars** class defines the characteristics of the command bar collection objects and describes the look and behavior of the collection objects through its properties and methods. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

You can customize your work environment by choosing which commands appear in the menus. You can do this by adding, removing, and renaming commands. You can also determine a command's placement in a menu. For further customization, you can rename and restore menus and menu commands, as well as change their shortcuts. This applies to the Menu Bar menus as well as pop-up menus that you access by right-clicking.

You have complete control over the position and content of toolbars. Using the mouse, you can resize or move toolbars anywhere inside the work area, and choose which toolbars you want displayed in the Application Window. You can add, remove, and rearrange toolbar commands (except in the Toolbox). You can also create your own custom toolbars containing the commands you use most often.

In addition, you can change the appearance of toolbar buttons. You can choose to view text on the toolbar button in place of an image, edit the text that appears on the toolbar button, as well as edit the actual images.

{button ,AL(`CLS_CommandBars')} **Related Topics**

CommandBars.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **CommandBar**

CommandBars

Description

The **Item** property returns a value associated with the index number of an command bar object in the **CommandBars** collection of CorelDRAW. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar, or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Each type of command bar in CorelDRAW is identified by an index number. `CommandBars(5)` refers to the fifth command bar in the **CommandBars** collection of CorelDRAW. The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `CommandBars.Item(5)` is the same as `CommandBars(5)` - they both reference the fifth command bar in the collection.

You must reference a command bar in the collection by passing its index number as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **CommandBars** collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each command bar.

Example

The following code example displays the name of the first command bar in the **CommandBars** collection of CorelDRAW:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Name
End Sub
```

{button ,AL(`CLS_CommandBars;FNC_Item')} **Related Topics**

CommandBars.Count

Property **Count** AS Long

CommandBars

Description

The **Count** property returns the number of command bar objects in the **CommandBars** collection of CorelDRAW. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Count** property returns a Read-Only value.

Example

The following code example counts the number of command bars in CorelDRAW:

```
Sub Bars()
MsgBox FrameWork.CommandBars.Count
End Sub
```

{button ,AL(`CLS_CommandBars;FNC_Count')} **Related Topics**

CommandBars.Add

Function **Add**(ByVal Name AS String, ByVal Position AS **cuiBarPosition**, ByVal Temporary AS Boolean) AS **CommandBar**

CommandBars

Description

The **Add** method adds a command bar object to the **CommandBars** collection in CorelDRAW. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

In order to use the **Add** method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter is a string value that identifies the name of a command bar. This value allows you to name the newly added command bar and is optional.
Position	The **Position** parameter allows identifies the position of the command bar. A command bar may be positioned at the top, bottom, left or right of the application window, as well as floating in the window. This value returns cuiBarPosition and is optional. The default value is cuiBarFloating (4).
Temporary	Lets you specify if the command bar is a permanent part of the CorelDRAW framework. This value is optional and the default value is False.

Example

The following code example adds a new command bar to the **CommandBars** collection in CorelDRAW. The new bar is called "Text" and it appears at the top of the framework:

```
Sub Bars()
Framework.CommandBars.Add ("Text", cuiBarTop, False)
End Sub
```

{button ,AL(`CLS_CommandBars;FNC_Add')} **Related Topics**

CommandBar properties

CommandBar Legend

- BuiltIn

- Controls
  - Enabled

- Height
  - Index
  - Left

- Modes

- Name
  - NameLocal
  - Position
  - Protection
  - Top

- Type
  - Visible

- Width

CommandBar methods

CommandBar Legend

- Delete
- Reset
- SetWidth
- ShowPopup

CommandBar

Class **CommandBar**

Properties Methods Referenced by

The **CommandBar** class defines the characteristics of the command bar objects and describes the look and behavior of the object through its properties and methods. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

You can customize your work environment by choosing which commands appear in the menus. You can do this by adding, removing, and renaming commands. You can also determine a command's placement in a menu. For further customization, you can rename and restore menus and menu commands, as well as change their shortcuts. This applies to the Menu Bar menus as well as pop-up menus that you access by right-clicking.

You have complete control over the position and content of toolbars. Using the mouse, you can resize or move toolbars anywhere inside the work area, and choose which toolbars you want displayed in the Application Window. You can add, remove, and rearrange toolbar commands (except in the Toolbox). You can also create your own custom toolbars containing the commands you use most often.

In addition, you can change the appearance of toolbar buttons. You can choose to view text on the toolbar button in place of an image, edit the text that appears on the toolbar button, as well as edit the actual images.

{button ,AL(`CLS_CommandBar')} **Related Topics**

CommandBar.Height

Property **Height** AS Long

CommandBar

Description

The **Height** property returns the height of a command bar control in CorelDRAW. The height is measured in document units.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Height** property returns a Read-Only value.

Example

The following code example displays the height of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Height
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Height')} **Related Topics**

CommandBar.Type

Property **Type** AS **cuiBarType**

CommandBar

Description

The **Type** property returns the type of a command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Type** property returns a Read-Only value of cuiBarType.

Example

The following code example displays the type of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Type
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Type')} **Related Topics**

cuiBarTypeNormal=0

cuiBarTypeMenuBar=1

cuiBarTypePopup=2

cuiBarTypeStatusBar=3

cuiBarTypePropertyBar=4

CommandBar.Visible

Property **Visible** AS Boolean

CommandBar

Description

The **Visible** property returns a True or False value that indicates the visibility status of a command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

If the **Visible** property is True, the command bar is visible in CorelDRAW.

Example

The following code example makes the first command bar in the **CommandBars** collection visible in CorelDRAW, if it is not already visible:

```
Sub Bars()
Framework.CommandBars.Item(1).Visible = True
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Visible')} **Related Topics**

CommandBar.Controls

Property **Controls** AS **CommandBarControls**

CommandBar

Description

The **Controls** property returns a value associated with the **Controls** collection contained within the command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Controls** property returns a Read-Only value.

Example

The following code example counts the number of controls in the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Controls.Count
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Controls')} **Related Topics**

CommandBar.BuiltIn

Property **BuiltIn** AS Boolean

CommandBar

Description

The **BuiltIn** property returns a True or False value that indicates if a control on a command bar is a built in system control in CorelDRAW. Build in system controls are commands that are mandatory components of CorelDRAW. For example, the Save command is a build in system control. If the **BuitIn** property is set to True, the command is a built in system control.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **BuiltIn** property returns a Read-Only value.

Example

The following code example specifies if the first command bar in the **CommandBars** collection is a built in command bar:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).BuiltIn
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_BuiltIn')} **Related Topics**

CommandBar.Enabled

Property **Enabled** AS Boolean

CommandBar

Description

The **Enabled** property returns a True or False value that indicates in a control on a command bar is currently available in CorelDRAW. A feature is enabled if it is visible and available for use within the application. Commands may be disabled as a result of a specific action within the application. If the **Enabled** property is True, the command is available for use in CorelDRAW.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example disables the first command bar in the **CommandBars** collection:

```
Sub Bars()
Framework.CommandBars.Item(1).Enabled = False
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Enabled')} **Related Topics**

CommandBar.Left

Property **Left** AS Long

CommandBar

Description

The **Left** property returns or set the screen position, in screen pixels, of the command bar's left edge in CorelDRAW. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the left position of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Left
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Left')} **Related Topics**

CommandBar.Top

Property **Top** AS Long

CommandBar

Description

The **Top** property returns or set the screen position, in screen pixels, of the command bars top edge in CorelDRAW. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the top position of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Top
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Top')} **Related Topics**

CommandBar.Modes

Property **Modes** As **CommandBarModes**

Description

The **Modes** property returns the mode of a command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

This property returns a Read-Only value.

Example

The following code example counts the number of modes in the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Modes.Count
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Modes')} **Related Topics**

CommandBar.Width

Property **Width** AS Long

CommandBar

Description

The **Width** property returns the width of a command bar control in CorelDRAW. The width is measured in document units.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Width** property returns a Read-Only value.

Example

The following code example displays the width of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Width
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Width')} **Related Topics**

CommandBar.Index

Property **Index** AS Long

CommandBar

Description

The **Index** property returns the index number of a command bar control in the control's collection in CorelDRAW. The index number uniquely identifies the control within the collection.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Index** property returns a Read-Only value.

Example

The following code example uses the index number of the first command bar to display the width of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Width
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Index')} **Related Topics**

CommandBar.Name

Property **Name** AS String

CommandBar

Description

The **Name** property returns a string value that uniquely identifies the name of a control in the control's collection in CorelDRAW.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the name of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Name
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Name')} **Related Topics**

CommandBar.NameLocal

Property **NameLocal** AS String

CommandBar

Description

The **NameLocal** property returns a string value that uniquely identifies the localized name of a control in the control's collection in CorelDRAW. A localized name

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the localized name of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).NameLocal
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_NameLocal')} **Related Topics**

cuiBarLeft=0

cuiBarTop=1

cuiBarRight=2

cuiBarBottom=3

cuiBarFloating=4

CommandBar.Position

Property **Position** AS **cuiBarPosition**

CommandBar

Description

The **Position** property returns or set the position of the command bar in CorelDRAW. A command bar can be positioned at the top, bottom, left or right of the application window. It can also be floating within the window.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Position** property returns a value of cuiBarPosition.

Example

The following code example displays the position of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Position
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Position')} **Related Topics**

cuiBarNoProtection=0

cuiBarNoCustomize=1

cuiBarNoResize=2

cuiBarNoMove=4

cuiBarNoChangeVisible=8

cuiBarNoChangeDock=16

cuiBarNoVerticalDock=32

cuiBarNoHorizontalDock=64

CommandBar.Protection

Property **Protection** AS **cuiBarProtection**

CommandBar

Description

The **Protection** property returns or set the protection level of the command bar in CorelDRAW. The protection level of a bar decides the level of editing and customization that can be applied to the command bar.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Protection** property returns a value of cuiBarProtection.

Example

The following code example displays the protection value of the first command bar in the **CommandBars** collection:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Protection
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Protection')} **Related Topics**

CommandBar.Delete

Sub **Delete**()

CommandBar

Description

The **Delete** method removes a control from the command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example deletes the first command bar in the **CommandBars** collection:

```
Sub Bars()
Framework.CommandBars.Item(1).Delete
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Delete')} **Related Topics**

cuiControlTypeControl=0

cuiControlTypeButton=1

cuiControlTypeComboBox=2

cuiControlTypePopup=3

cuiControlTypeLabel=4

CommandBar.Reset

Sub **Reset**()

CommandBar

Description

The **Reset** method restores a control's default property values in the command bar. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example resets the first command bar in the **CommandBars** collection:

```
Sub Bars()
Framework.CommandBars.Item(1).Reset
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_Reset')} **Related Topics**

CommandBar.ShowPopup

Sub **ShowPopup**(ByVal **X** AS Variant, ByVal **Y** AS Variant)

CommandBar

Description

The **ShowPopup** method shows the command bar as a popup bar in CorelDRAW. A popup command bar appears as needed in the application window. It is not a fixed bar that is docked in the window. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

In order to use the **ShowPopup** method, you must pass the **X** and **Y** values as parameters:

Parameters	Description
X	The X parameter specifies the horizontal position of the popup command bar. The X parameter is measured in document units.cdrUnit
Y	The Y parameter specifies the vertical position of the popup command bar. The X parameter is measured in document units.cdrUnit

Example

The following code example displays the first command bar in the **CommandBars** collection as a pop up command bar, at the coordinate (2,2):

```
Sub Bars()
Framework.CommandBars.Item(1).ShowPopup (2, 2)
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_ShowPopup')} **Related Topics**

CommandBar.SetWidth

Sub **SetWidth**(ByVal **Width** AS Long)

CommandBar

Description

The **SetWidth** method sets the width of the command bar in CorelDRAW. The width of the bar is measured in document units. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

In order to use the **SetWidth** method, you must pass the width value as a parameter:

Parameters	Description
Width	The **Width** parameter specifies a numeric value that sets the width of the command bar. The **Width** value is measured in document units.

Example

The following code example sets the width of the first command bar in the **CommandBars** collection:

```
Sub Bars()
Framework.CommandBars.Item(1).SetWidth = 2
End Sub
```

{button ,AL(`CLS_CommandBar;FNC_SetWidth')} **Related Topics**

CommandBarControls properties

CommandBarControls Legend

- Count

- Item

CommandBarControls

Class **CommandBarControls**

Properties Referenced by

The **CommandBarControls** class defines the characteristics of the command bar control collection objects and describes the look and behavior of the collection objects through its properties and methods.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

You can customize your work environment by choosing which commands appear in the menus. You can do this by adding, removing, and renaming commands. You can also determine a command's placement in a menu. For further customization, you can rename and restore menus and menu commands, as well as change their shortcuts. This applies to the Menu Bar menus as well as pop-up menus that you access by right-clicking.

You have complete control over the position and content of toolbars. Using the mouse, you can resize or move toolbars anywhere inside the work area, and choose which toolbars you want displayed in the Application Window. You can add, remove, and rearrange toolbar commands (except in the Toolbox). You can also create your own custom toolbars containing the commands you use most often.

In addition, you can change the appearance of toolbar buttons. You can choose to view text on the toolbar button in place of an image, edit the text that appears on the toolbar button, as well as edit the actual images.

{button ,AL(`CLS_CommandBarControls')} **Related Topics**

CommandBarControls.Count

Property **Count** AS Long

CommandBarControls

Description

The **Count** property returns the number of command bar control objects in the **CommandBarControls** collection of CorelDRAW. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar, or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **Count** property returns a Read-Only value.

Example

The following code example counts the number of controls in the first command bar of the **CommandBars** collection:

```
Sub Control()
MsgBox FrameWork.CommandBars(1).Controls.Count
End Sub
```

{button ,AL(`CLS_CommandBarControls;FNC_Count')} **Related Topics**

CommandBarControls.Item

Property **Item**(ByVal **Index** AS Long) AS **CommandBarControl**

CommandBarControls

Description

The **Item** property returns a value associated with the index number of a command bar control object in the **CommandBarControls** collection of CorelDRAW. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar, or a properties bar. Each contains specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Each type of command bar in CorelDRAW is identified by an index number. `CommandBarControls(5)` refers to the fifth command bar control in the **CommandBarControls** collection of CorelDRAW. The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `CommandBarControls.Item(5)` is the same as `CommandBarControls(5)` - they both reference the fifth command bar control in the collection.

You must reference a command bar in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **CommandBarControls** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the ID number of the first control in the first command bar in the **CommandBars** collection of CorelDRAW:

```
Sub Bars()
MsgBox Framework.CommandBars.Item(1).Controls(1).ID
End Sub
```

{button ,AL(`CLS_CommandBarControls;FNC_Item')} **Related Topics**

CommandBarControl properties

CommandBarControl Legend

- Caption
  - DescriptionText
  - Height
- ID
  - Parameter
  - Tag
  - ToolTipText
  - Visible
  - Width

CommandBarControl

Class **CommandBarControl**

Properties Referenced by

The **CommandBarControl** class defines the characteristics of the command bar control objects and describes the look and behavior of the objects through its properties and methods.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

You can customize your work environment by choosing which commands appear in the menus. You can do this by adding, removing, and renaming commands. You can also determine a command's placement in a menu. For further customization, you can rename and restore menus and menu commands, as well as change their shortcuts. This applies to the Menu Bar menus as well as pop-up menus that you access by right-clicking.

You have complete control over the position and content of toolbars. Using the mouse, you can resize or move toolbars anywhere inside the work area, and choose which toolbars you want displayed in the Application Window. You can add, remove, and rearrange toolbar commands (except in the Toolbox). You can also create your own custom toolbars containing the commands you use most often.

In addition, you can change the appearance of toolbar buttons. You can choose to view text on the toolbar button in place of an image, edit the text that appears on the toolbar button, as well as edit the actual images.

{button ,AL(`CLS_CommandBarControl')} **Related Topics**

CommandBarControl.Caption

Property **Caption** AS String

CommandBarControl

Description

The **Caption** property returns or sets the caption of a control in a command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the caption of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).Caption
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_Caption')} **Related Topics**

CommandBarControl.DescriptionText

Property **DescriptionText** AS String

CommandBarControl

Description

The **DescriptionText** property returns or sets a string value that describes a control in a command bar in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the descriptive text of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).DescriptionText
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_DescriptionText')} **Related Topics**

CommandBarControl.Height

Property **Height** AS Long

CommandBarControl

Description

The **Height** property returns or sets the height of a command bar control in CorelDRAW. The height is measured in document units.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the height of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).Height
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_Height')} **Related Topics**

CommandBarControl.Width

Property **Width** AS Long

CommandBarControl

Description

The **Width** property returns or sets the width of a command bar control in CorelDRAW. The height is measured in document units.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the width of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).Width
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_Width')} **Related Topics**

CommandBarControl.ID

Property **ID** AS Long

CommandBarControl

Description

The **ID** property returns a numerical value that uniquely identifies a command bar control in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

The **ID** property returns a Read-Only value.

Example

The following code example displays the ID of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).ID
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_ID')} **Related Topics**

CommandBarControl.Parameter

Property **Parameter** AS Variant

CommandBarControl

Description

The **Parameter** property returns or sets a parameter for a command bar control in a command bar of CorelDRAW. A parameter is a user customizable piece of variant data.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

{button ,AL(`CLS_CommandBarControl;FNC_Parameter')} **Related Topics**

CommandBarControl.Tag

Property **Tag** AS String

CommandBarControl

Description

The **Tag** property returns a user-defined string value that describes a command bar control or group of controls in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the tag value of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).Tag
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_Tag')} **Related Topics**

CommandBarControl.ToolTipText

Property **ToolTipText** AS String

CommandBarControl

Description

The **ToolTipText** property returns or sets a string value that appears in a tool tip for the command bar control in CorelDRAW. A tool tip describes individual features in the application, whereas the CorelDRAW tutorial guides you through basic tutorial procedures as you complete a range of tasks. The Online Hints window provides you with information about the tool you're using, or the possible tasks or actions you can perform.

A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

Example

The following code example displays the tool tip text of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).ToolTipText
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_ToolTipText')} **Related Topics**

CommandBarControl.Visible

Property **Visible** AS Boolean

CommandBarControl

Description

The **Visible** property returns a True or False value that indicates the visibility status of a command bar control in CorelDRAW. A command bar is a customized menu or toolbar that enables you quickly access to commands in the application. A command bar can be a toolbar, menu bar, status bar, pop-up menu bar or a properties bar. Each contain specific commands that customize a drawing in CorelDRAW.

A command is a word or control that initiates an action when selected or clicked. Commands can be accessed from a menu or by clicking buttons on a toolbar. A command button is a button in a dialog box or toolbar that is used to carry out an action, such as resetting values or opening a dialog box.

If the **Visible** property is True, the command bar control is visible in CorelDRAW.

Example

The following code example displays the visibility status of the first control in the first command bar in the **CommandBars** collection in a message box:

```
Sub Bars()
MsgBox Framework.CommandBars(1).Controls(1).Visible
End Sub
```

{button ,AL(`CLS_CommandBarControl;FNC_Visible')} **Related Topics**

cuiLabelText=0

cuiLabelImage=1

A file that contains lists of shortcut keys. Shortcut keys are used to speed up, or "accelerate," editing tasks. Different tables are active depending on what you're doing. For example, when you highlight text the Text Editing accelerator table becomes active. If no text is selected, the Main accelerator table is active.

Bitmap properties

Bitmap Legend

- ExternallyLinked
- ResolutionX
- ResolutionY
- SizeHeight
- SizeWidth

Bitmap methods

Bitmap Legend

- Inflate
- ResolveLink
- UpdateLink

Bitmap

Class **Bitmap**

Properties Methods Referenced by

The **Bitmap** class defines the characteristics of bitmap objects and describes the look and behavior of the objects through its properties and methods. A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Vectors are defined mathematically as a series of points joined by lines. Unlike bitmaps, vectors represent shapes as a series of lines and curves that you can resize without loss of quality. Graphical elements in a vector file are called objects. Each object is a self-contained entity with properties such as color, shape, outline, and size included in its definition. For more information about vectors, see "CorelDRAW Concepts".

Because a bitmap is a collection of arranged pixels, you cannot manipulate its parts individually. The color and shape appear continuous when viewed from a distance. However, CorelDRAW lets you manipulate bitmaps in different ways. You can crop bitmaps to decrease the visible area and reduce the size of files by linking bitmaps to your drawings. You can also trace bitmap images, which converts them to vector images that are easier to manipulate.

CorelDRAW also lets you change an image by manipulating colors, tones, and resampling. For example, you can hide or show bitmap colors, adjust image tone, or resample to change image size or resolution. You can inflate a bitmap to make an effect cover the entire image. You can convert an image from vectors to bitmaps, or from one color mode to another.

CorelDRAW provides special effects that you can apply to images. For example, you can create three-dimensional images, mimic artistic styles, blur or clear your images, and manipulate image colors, contours, noise, distortions, and sharpness. For more information on applying special effects to bitmaps, see "Applying special effects to bitmaps."

{button ,AL(`CLS_Bitmap')} **Related Topics**

Bitmap.SizeWidth

Property **SizeWidth** AS Long

Bitmap

Description

The **SizeWidth** property returns the width of a bitmap in CorelDRAW. A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

This property returns a Read-Only value.

Example

The following code example displays the width and height of a bitmap object in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    MsgBox .SizeWidth
    MsgBox .SizeHeight
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_SizeWidth')} **Related Topics**

Bitmap.SizeHeight

Property **SizeHeight** AS Long

Bitmap

Description

The **SizeHeight** property returns the height of a bitmap in CorelDRAW. A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

This property returns a Read-Only value.

Example

The following code example displays the width and height of a bitmap object in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    MsgBox .SizeWidth
    MsgBox .SizeHeight
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_SizeHeight')} **Related Topics**

Bitmap.ResolutionX

Property **ResolutionX** AS Long

Bitmap

Description

The **ResolutionX** property returns the horizontal resolution of a bitmap in CorelDRAW. Resolution refers to the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

This property returns a Read-Only value.

Example

The following code example displays the horizontal and vertical resolution of a bitmap object in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    MsgBox .ResolutionX
    MsgBox .ResolutionY
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_ResolutionX')} **Related Topics**

Bitmap.ResolutionY

Property **ResolutionY** AS Long

Bitmap

Description

The **ResolutionY** property returns the vertical resolution of a bitmap in CorelDRAW. Resolution refers to the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

This property returns a Read-Only value.

Example

The following code example displays the horizontal and vertical resolution of a bitmap object in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    MsgBox .ResolutionX
    MsgBox .ResolutionY
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_ResolutionY')} **Related Topics**

Bitmap.ExternallyLinked

Property **ExternallyLinked** AS Boolean

Bitmap

Description

The **ExternallyLinked** property returns a True or False value that indicates if a bitmap is linked externally outside of the document or if it is embedded within the document. An externally linked bitmap is a bitmap that is not embedded in a document, but is referenced by the application. A lower-resolution "placeholder" image is visible in the document.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

This property returns a Read-Only value.

Example

The following code example displays the link status of a bitmap in CorelDRAW:

```
Sub Link()
MsgBox ActiveShape.Bitmap.ExternallyLinked
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_ExternallyLinked')} **Related Topics**

Bitmap.ResolveLink

Sub **ResolveLink**()

Bitmap

Description

The **ResolveLink** method embeds an externally linked bitmap in CorelDRAW. An externally linked bitmap is a bitmap that is not embedded in a document, but is referenced by the application. A lower-resolution "placeholder" image is visible in the document.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Example

The following code example embeds the active bitmap if it is not already embedded in CorelDRAW:

```
Sub Embed()
With ActiveShape.Bitmap
    If .ExternallyLinked = True Then
    .ResolveLink
    End If
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_ResolveLink')} **Related Topics**

Bitmap.UpdateLink

Sub **UpdateLink**()

Bitmap

Description

The **UpdateLink** method updates an externally linked bitmap from its source and applies the updates to the linked bitmap in CorelDRAW. An externally linked bitmap is a bitmap that is not embedded in a document, but is referenced by the application. A lower-resolution "placeholder" image is visible in the document.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Example

The following code example updates the active bitmap if it is externally linked:

```
Sub Embed()
With ActiveShape.Bitmap
    If .ExternallyLinked = True Then
    .UpdateLink
    End If
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_UpdateLink')} **Related Topics**

Bitmap.Inflate

Sub **Inflate**(ByVal **Width** AS Long, ByVal **Height** AS Long)

Bitmap

Description

The **Inflate** method increases the size of a bitmap in CorelDRAW by increasing its height and width values. A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Parameters	Description
Width	Sets the horizontal size of a bitmap, in document units.
Height	Sets the vertical size of a bitmap, in document units.

Example

The following code example increases the size of the active bitmap in CorelDRAW:

```
Sub Size()
With ActiveShape.Bitmap
    .Inflate 5, 5
End With
End Sub
```

{button ,AL(`CLS_Bitmap;FNC_Inflate')} **Related Topics**

Text properties

Text Legend

- AlignProperties
- AlignPropertiesInRange
- CharacterCount
- Characters
- Contents
- FontProperties
- FontPropertiesInRange
- FramesInLink
- HyphenationSettings
- HyphenationSettingsInRange
- IsHTMLCompatible
- LineCount
- Overflow
- ParagraphCount
- Paragraphs
- SpaceProperties
- SpacePropertiesInRange
- Type
- UnusedFramesInLink
- WordCount
- Words

Text methods

Text Legend

- ConvertToArtistic
- ConvertToParagraph
- ExportToFile
- Find
- FitToPath
- ImportFromFile
- LinkToFrame
- MakeHTMLCompatible
- Replace

Text

Class **Text**

Properties Methods Referenced by

The **Text** class defines the characteristics of text objects and describes the look and behavior of the objects through its properties and methods. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool.

Clicking in the Drawing Window and typing creates Artistic text. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. An Artistic text object can contain up to 32,000 characters. CorelDRAW automatically applies the default Artistic text style, which you can change using the Styles Manager.

Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. Paragraph formatting features let you flow text between frames and columns, create bulleted lists, set tabs and indents, and add drop caps. CorelDRAW automatically applies the default Paragraph text style, which you can change using the Styles Manager.

Some formatting features apply to both Artistic text and Paragraph text. For example, spacing text and applying bold formatting. Other formatting features apply only to Paragraph text. For example, adding columns and drop caps.

While working with text, you can use the writing tools to verify grammar and spelling, replace a word with a synonym, create a user word list for the writing tools to use, or generate a count of text elements.

There are two types of Paragraph text frames: fixed-sized and automatically sized. When you add a frame of a fixed size, the size of the frame you draw doesn't change. If you type more text than the frame can hold, the frame size doesn't change, and the text is cut off. If you choose to add a frame that sizes automatically, the frame size adjusts vertically according to the amount of text you type.

After you create a text object you can convert the text type. For example, you can convert Artistic text to Paragraph text and Paragraph text can be converted to Artistic text.

When you're working with Artistic text, you can modify the shape of characters. Before you do this, you must convert the text to curves. By converting, you can transform characters to single line and curve objects. Then you can use the Shape tool to add, delete, or move the nodes comprising a character to alter its shape.

Before performing any operation to text, you need to select the text. The tool you use to select a character, line, or paragraph varies with the operation. When you select a text object, eight selection handles and an X appears in the center of the object. By clicking the center X, you can transform, apply special effects, and make global formatting changes to whole text objects.

You can also add text using the Clipboard. When you use the Special Paste command, you can insert text as Paragraph text, as Artistic text, or as an OLE object, provided the originating application is OLE compliant and it is open.

{button ,AL(`CLS_Text')} **Related Topics**

Text styles are variations in a typeface or font. Some common styles include Roman (regular or normal), bold, italic, and bold italic.

Text.CharacterCount

Property **CharacterCount** AS Long

Text

Description

The **CharacterCount** property returns a value associated with number of characters in a text string in CorelDRAW. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

For example, in the text string "CorelDRAW Help", the character count is 14. Even though there are only 13 text characters, the space between "CorelDRAW" and "Help" is counted as a character.

It is important to note that carriage returns in paragraph text are not counted when using the **CharacterCount** property.

The **CharacterCount** property returns a Read-Only value.

Example

The following code example displays the number of characters in the text string "CorelDRAW Help", using the **CharacterCount** property:

```
Sub TextCount()
With ActiveShape.Text
    MsgBox .CharacterCount
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_CharacterCount')} **Related Topics**

Text.WordCount

Property **WordCount** AS Long

Text

Description

The **WordCount** property returns a value associated with number of words in a text string in CorelDRAW. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

For example, in the text string "CorelDRAW Help", the word count is 2.

The **WordCount** property returns a Read-Only value.

Example

The following code example displays the number of words in the text segment "CorelDRAW Help", using the **WordCount** property:

```
Sub TextWord()
With ActiveShape.Text
    MsgBox .WordCount
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_WordCount')} **Related Topics**

Text.LineCount

Property **LineCount** AS Long

Text

Description

The **LineCount** property returns a value associated with number of lines in a text string in CorelDRAW. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

A line can be a continuation of wrapped text from a previous line or it may be a new line created by a carriage return. Both line types are counted with the **LineCount** property.

The **LineCount** property returns a Read-Only value.

Example

The following code example counts the number of lines in a text string in the active shape, using the **LineCount** property:

```
Sub TextLine()
With ActiveShape.Text
    MsgBox .LineCount
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_LineCount')} **Related Topics**

Text.ParagraphCount

Property **ParagraphCount** AS Long

Text

Description

The **ParagraphCount** property returns a value associated with number of paragraphs in a text string in CorelDRAW. Paragraph text is created with the Text tool and is used to add large blocks of text for ads, brochures, and other text-intensive projects. Paragraph formatting features let you flow text between frames and columns, create bulleted lists, set tabs and indents, and add drop caps. CorelDRAW automatically applies the default Paragraph text style, which you can change using the Styles Manager.

Unlike lines, which can contain wrapped text from the previous line, a paragraph is marked by a carriage return, which signals the end of one paragraph and the beginning of another paragraph.

The **ParagraphCount** property returns a Read-Only value.

Example

The following code example displays the number of paragraphs in the text string of the active shape, using the **ParagraphCount** property:

```
Sub TextParaCount()
With ActiveShape.Text
    MsgBox .ParagraphCount
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_ParagraphCount')} **Related Topics**

Text.Type

Property **Type** AS **cdrTextType**

Text

Description

The **Type** property returns the text type of a text string in CorelDRAW. The text type can be artistic or paragraph. In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

Text can be of artistic, paragraph, fitted artistic, or fitted paragraph type. The **Type** property returns a value of cdrTextType.

The **Type** property returns a Read-Only value.

Example

The following code example displays the text type of the text string in the active shape, using the **Type** property:

```
Sub TextType()
With ActiveShape.Text
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Type')} **Related Topics**

cdrArtisticText=0

cdrParagraphText=1

cdrArtisticFittedText=2

cdrParagraphFittedText=3

Text.FramesInLink

Property **FramesInLink** AS Long

Text

Description

The **FramesInLink** property returns a value associated with the number of linked paragraph text frames in CorelDRAW. A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.

Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink or enlarge a linked frame, or change the size of the text, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

The **FramesInLink** property returns a Read-Only value.

Example

The following code example displays the number of linked paragraph text frame objects in the active shape, using the **FramesInLink** property:

```
Sub TextFrameLink()
With ActiveShape.Text
    MsgBox .FramesInLink
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_FramesInLink')} **Related Topics**

Text.UnusedFramesInLink

Property **UnusedFramesInLink** AS Long

Text

Description

The **UnusedFramesInLink** property returns a value associated with the number of unused linked paragraph text frames in CorelDRAW. A frame is unused if it is linked with other text frames but contains no text. A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.

Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink or enlarge a linked frame, or change the size of the text, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

The **UnusedFramesInLink** property returns a Read-Only value.

Example

The following code example displays the number of unused linked paragraph text frames in the active shape, using the **UnusedFramesInLink** property:

```
Sub TextUnusedFrame()
With ActiveShape.Text
    MsgBox .UnusedFramesInLink
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_UnusedFramesInLink')} **Related Topics**

Text.Overflow

Property **Overflow** AS Boolean

Text

Description

The **Overflow** property returns a True or False value indicating if the amount of text exceeds the space available in a chain of linked paragraph text frames in CorelDRAW.. A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink or enlarge a linked frame, or change the size of the text, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

For example, if you have two linked text frames and you type more text than the frames can contain, the **Overflow** is set to True. The text remains part of the active shape, but it will not appear unless the frames are adjusted to fit the text, or until an additional frame is linked to contain the overflow of text.

The **Overflow** property returns a Read-Only value.

Example

The following code example displays a True of False value indicating the presence of overflow text in the active shape, using the **Overflow** property:

```
Sub TextOverflow()
With ActiveShape.Text
    MsgBox .Overflow
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Overflow')} **Related Topics**

Text.FontProperties

Property **FontProperties**(ByVal Frames AS **cdrTextFrames**) AS **StructFontProperties**

Text

Description

The **FontProperties** property returns or sets a value associated with the font of text in a selected frame in CorelDRAW. A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of fonts that can be set using the **FontProperties** property. For a full listing of properties, see StructFontProperties.

In order to set or return font properties in a selected text frame, you must pass the selected frame in the **Frames** parameter:

Parameters	Description
Frames	**Frames** refers to a text frame, the rectangle that contains a block of Paragraph text created using the Text tool. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow. The **Frames** parameter has a value of cdrTextFrames.The default **Frames** value is cdrAllFrames (2).

Example

The following code example sets the text in all linked text frames with uppercase font, using the **FontProperties** property. The **Uppercase** property returns a value of cdrFontCase:

```
Sub TextFont()
With ActiveShape.Text
    .FontProperties(cdrAllFrames).Uppercase = cdrAllCapsFontCase
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_FontProperties')} **Related Topics**

cdrNormalFontCase=0

cdrSmallCapsFontCase=1

cdrAllCapsFontCase=2

cdrMixedFontCase=3

cdrThisFrameOnly=0

cdrStartAtThisFrame=1

cdrAllFrames=2

Text.FontPropertiesInRange

Property **FontPropertiesInRange**(ByVal **StartIndex** AS Long, ByVal **Count** AS Long, ByVal IndexingType AS **cdrTextIndexingType**) AS **StructFontProperties**

Text

Description

The **FontPropertiesInRange** property returns or sets a value associated with the font in a range of text in a selected frame in CorelDRAW. When you define the **StartIndex**, **Count**, and **IndexingType** parameters, you define a range of text in which to apply specific font properties.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of fonts that can be set using the **FontPropertiesInRange** property. For a full listing of properties, see **StructFontProperties.**

In order to set or return font properties in a selected text frame, you must pass the **StartIndex**, **Count**, and **IndexingType** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame.
Count	**Count** defines how many times the font property is applied to text within a range. If the index type is Word, the start index is 3 and the count is 2, the **Count** parameter applies the font property twice, to the third and fourth words in the selected text frame.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example applies a single thick underline to the second, third and fourth words in the selected text frame, using the **FontPropertiesInRange** property. The **Underline** property returns a value of cdrFontLine:

```
Sub TextFontRange()
With ActiveShape.Text
    .FontPropertiesInRange(2, 3, cdrWordIndexing).Underline = cdrSingleThickWordFontLine
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_FontPropertiesInRange')} **Related Topics**

cdrCharacterIndexing=0

cdrWordIndexing=1

cdrParagraphIndexing=2

cdrNoFontLine=0

cdrSingleThinFontLine=1

cdrSingleThinWordFontLine=2

cdrSingleThickFontLine=3

cdrSingleThickWordFontLine=4

cdrDoubleThinFontLine=5

cdrDoubleThinWordFontLine=6

cdrMixedFontLine=7

Text.AlignProperties

Property **AlignProperties**(ByVal Frames AS **cdrTextFrames**) AS **StructAlignProperties**

Text

Description

The **AlignProperties** property returns or sets a value associated with the alignment of text in a selected frame in CorelDRAW. Alignment changes the position of text within a selected text frame. For example, text can be aligned to the left, right or center of a text frame.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of text alignment that can be set using the **AlignProperties** property. For a full listing of properties, see **StructAlignProperties**.

In order to set or return alignment properties in a selected text frame, you must pass the selected frame in the **Frames** parameter:

Parameters	Description
Frames	**Frames** refers to a text frame, the rectangle that contains a block of Paragraph text created using the Text tool. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow. The **Frames** parameter has a value of cdrTextFrames.The default **Frames** value is cdrAllFrames (2).

Example

The following code example right aligns all text in the selected text frame only, using the **AlignProperties** property. The **Alignment** property returns a value of cdrAlignment:

```
Sub TextAlign()
With ActiveShape.Text
    .AlignProperties(cdrThisFrameOnly).Alignment = cdrRightAlignment
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_AlignProperties')} **Related Topics**

cdrNoAlignment=0

cdrLeftAlignment=1

cdrRightAlignment=2

cdrCenterAlignment=3

cdrFullJustifyAlignment=4

cdrForceJustifyAlignment=5

cdrMixedAlignment=6

Text.AlignPropertiesInRange

Property **AlignPropertiesInRange**(ByVal **StartIndex** AS Long, ByVal **Count** AS Long, ByVal IndexingType AS **cdrTextIndexingType**) AS **StructAlignProperties**

Text

Description

The **AlignPropertiesInRange** property returns or sets a value associated with the alignment in a range of text in a selected frame in CorelDRAW. When you define the **StartIndex**, **Count**, and **IndexingType** parameters, you define a range of text in which to apply specific alignment properties. Alignment changes the position of text within a selected text frame. For example, text can be aligned to the left, right or center of a text frame.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of alignment that can be set using the **AlignPropertiesInRange** property. For a full listing of properties, see **StructAlignProperties**.

In order to set or return alignment properties in a selected text frame, you must pass the **StartIndex**, **Count**, and **IndexingType** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame.
Count	**Count** defines how many times the alignment property is applied to text within a range. If the index type is Word, the start index is 3 and the count is 2, the **Count** parameter applies the alignment property twice, to the third and fourth words in the selected text frame.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example right aligns the text in the second and third paragraphs in the selected text frame, using the **AlignPropertiesInRange** property. The **Alignment** property returns a value of cdrAlignment:

```
Sub TextAlignRange()
With ActiveShape.Text
    .AlignPropertiesInRange(2, 2, cdrParagraphIndexing).Alignment = cdrRightAlignment
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_AlignPropertiesInRange')} **Related Topics**

Text.SpaceProperties

Property **SpaceProperties**(ByVal Frames AS **cdrTextFrames**) AS **StructSpaceProperties**

Text

Description

The **SpaceProperties** property returns or sets a value associated with the spacing of text in a selected text frame in CorelDRAW. Text has spaces between characters, words, lines, and paragraphs. Any change to text spacing affects the appearance of the text in a text frame.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of text spacing that can be set using the **SpaceProperties** property. For a full listing of properties, see **StructSpaceProperties**.

In order to set or return spacing properties in a selected text frame, you must pass the selected frame in the **Frames** parameter:

Parameters	Description
Frames	**Frames** refers to a text frame, the rectangle that contains a block of Paragraph text created using the Text tool. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow. The **Frames** parameter has a value of cdrTextFrames.The default **Frames** value is cdrAllFrames (2).

Example

The following code example doubles the normal word spacing in the text of the selected text frame, using the **SpaceProperties** property:

```
Sub TextSpace()
With ActiveShape.Text
    .SpaceProperties(cdrThisFrameOnly).WordSpacing = 200    '200% - doubles spacing
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_SpaceProperties')} **Related Topics**

Text.SpacePropertiesInRange

Property **SpacePropertiesInRange**(ByVal **StartIndex** AS Long, ByVal **Count** AS Long, ByVal IndexingType AS **cdrTextIndexingType**) AS **StructSpaceProperties**

Text

Description

The **SpacePropertiesInRange** property returns or sets a value associated with the spacing in a range of text in a selected text frame in CorelDRAW. When you define the **StartIndex**, **Count**, and **IndexingType** parameters, you define a range of text in which to apply specific spacing properties. Text has spaces between characters, words, lines, and paragraphs. Any change to text spacing affects the appearance of the text in a text frame.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of text spacing that can be set using the **SpacePropertiesInRange** property. For a full listing of properties, see **StructSpaceProperties**.

In order to set or return alignment properties in a selected text frame, you must pass the **StartIndex**, **Count**, and **IndexingType** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame.
Count	**Count** defines how many times the property is applied to text within a range. If the index type is Word, the start index is 3 and the count is 2, the **Count** parameter applies the property twice, to the third and fourth words in the selected text frame.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example sets the spacing after the first and second paragraphs of text in the selected text frame to 200%, using the **SpacePropertiesInRange** property. This doubles the default after-paragraph space setting:

```
Sub TextSpaceRange()
With ActiveShape.Text
    .SpacePropertiesInRange(1, 2, cdrParagraphIndexing).AfterParagraphSpacing = 200
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_SpacePropertiesInRange')} **Related Topics**

Text.HyphenationSettings

Property **HyphenationSettings**(ByVal Frames AS **cdrTextFrames**) AS **StructHyphenationSettings**

Text

Description

The **HyphenationSettings** property returns or sets a value associated with the hyphenation of text in a selected text frame in CorelDRAW. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of text hyphenation that can be set using the **HyphenationSettings** property. For a full listing of properties, see **StructHyphenationSettings**.

In order to set or return hyphenation properties in a selected text frame, you must pass the selected frame in the **Frames** parameter:

Parameters	Description
Frames	**Frames** refers to a text frame, the rectangle that contains a block of Paragraph text created using the Text tool. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow. The **Frames** parameter has a value of cdrTextFrames.The default **Frames** value is cdrAllFrames (2).

Example

The following code example enables automatic hyphenation of text in the selected text frame, using the **HyphenationSettings** property. By enabling hyphenation, CorelDRAW automatically divides words at the end of lines instead of wrapping them to the next line.

```
Sub TextHyphen()
With ActiveShape.Text
    .HyphenationSettings(cdrThisFrameOnly).UseAutomaticHyphenation = True
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_HyphenationSettings')} **Related Topics**

Text.HyphenationSettingsInRange

Property **HyphenationSettingsInRange**(ByVal **StartIndex** AS Long, ByVal **Count** AS Long, ByVal IndexingType AS **cdrTextIndexingType**) AS **StructHyphenationSettings**

Text

Description

The **HyphenationSettingsInRange** property returns or sets a value associated with the hyphenation in a range of text in a selected text frame in CorelDRAW. When you define the **StartIndex**, **Count**, and **IndexingType** parameters, you define a range of text in which to apply specific hyphenation properties. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

There are several properties of text hyphenation that can be set using the **HyphenationSettingsInRange** property. For a full listing of properties, see **StructHyphenationSettings**.

In order to set or return alignment properties in a selected text frame, you must pass the **StartIndex**, **Count**, and **IndexingType** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame.
Count	**Count** defines how many times a property is applied to text within a range. If the index type is Word, the start index is 3 and the count is 2, the **Count** parameter applies the property twice, to the third and fourth words in the selected text frame.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example enables automatic hyphenation of text in the first 20 words of text in the selected text frame, using the **HyphenationSettingsInRange** property. By enabling hyphenation, CorelDRAW automatically divides words at the end of lines instead of wrapping them to the next line.

```
Sub TextHyphen()
With ActiveShape.Text
    .HyphenationSettingsInRange(1, 20, cdrWordIndexing).UseAutomaticHyphenation = True
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_HyphenationSettingsInRange')} **Related Topics**

Text.Contents

Property **Contents**(ByVal Frames AS **cdrTextFrames**) AS String

Text

Description

The **Contents** property returns or replaces all text within a specified text frame in CorelDRAW. Even though text may change when using the **Contents** property, the text formatting within a text frame still applies to the new text. For example, if old text in a text frame is right aligned, the new text will also be right aligned in the text frame.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style..

In order to set or return content properties in a selected text frame, you must pass the selected frame in the **Frames** parameter:

Parameters	Description
Frames	**Frames** refers to a text frame, the rectangle that contains a block of Paragraph text created using the Text tool. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow. The **Frames** parameter has a value of cdrTextFrames.The default **Frames** value is cdrAllFrames (2).

Example

The following code example displays the text in all linked text frames, using the **Contents** property. Only one frame has to be selected - the **Contents** property takes all text from all linked text frames:

```
Sub TextContents()
With ActiveShape.Text
    MsgBox .Contents(cdrAllFrames)
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Contents')} **Related Topics**

Text.Characters

Property **Characters**(ByVal **StartIndex** AS Long, ByVal Count AS Long) AS String

Text

Description

The **Characters** property returns or replaces a value associated with a range of text characters in CorelDRAW. When you define the **StartIndex** and **Count** parameters, you define a range of text in which to apply specific properties.

In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

For example, in the text string "CorelDRAW Help", the character count is 14. Even though there are only 13 text characters, the space between "CorelDRAW" and "Help" is counted as a character.

It is important to note that carriage returns in paragraph text are not counted as characters when using the **Characters** property.

In order to set or replace characters in a selected text frame, you must pass the **StartIndex** and **Count** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a character range. If the start index is 3, the **StartIndex** parameter begins the range at the third text character in the selection.
Count	**Count** defines how many times the property is applied to character text within a range. If the start index is 3 and the count is 2, the **Count** parameter applies the property twice, to the third and fourth characters in the selection.

Example

The following code example displays the first three characters of text in the selected text frame, using the **Characters** property:

```
Sub TextCharacters()
With ActiveShape.Text
    MsgBox .Characters(1, 3)          'starts at the first character in the frame, includes the
first 3 characters
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Characters')} **Related Topics**

Text.Words

Property **Words**(ByVal **StartIndex** AS Long, ByVal Count AS Long) AS String

Text

Description

The **Words** property returns or replaces a value associated with a range of words in a text selection in CorelDRAW. When you define the **StartIndex** and **Count** parameters, you define a range of text in which to apply specific properties.

In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

For example, in the text string "CorelDRAW Help", the word count is 2.

In order to set or replace words in a selected text frame, you must pass the **StartIndex** and **Count** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a word range. If the start index is 3, the **StartIndex** parameter begins the range at the third word in the text selection.
Count	**Count** defines how many times the property is applied to word within a range. If the start index is 3 and the count is 2, the **Count** parameter applies the property twice, to the third and fourth words in the selection.

Example

The following code example replaces the first 2 words of text in the selected text frame with "CorelDRAW Help", using the **Words** property:

```
Sub TextWords()
With ActiveShape.Text
    .Words(1, 2) = "CorelDRAW Help"
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Words')} **Related Topics**

Text.Paragraphs

Property **Paragraphs**(ByVal **StartIndex** AS Long, ByVal Count AS Long) AS String

Text

Description

The **Paragraphs** property returns or replaces a value associated with a range of paragraphs in a text selection in CorelDRAW. When you define the **StartIndex** and **Count** parameters, you define a range of text in which to apply specific properties.

In CorelDRAW, you create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

Unlike lines, which can contain wrapped text from the previous line, a paragraph is marked by a carriage return, which signals the end of one paragraph and the beginning of another paragraph.

In order to set or replace paragraphs in a selected text frame, you must pass the **StartIndex** and **Count** values as parameters:

Parameters	Description
StartIndex	**StartIndex** defines the first text object in a paragraph range. If the start index is 3, the **StartIndex** parameter begins the range at the third paragraph in the text selection.
Count	**Count** defines how many times the property is applied to paragraph within a range. If the start index is 3 and the count is 2, the **Count** parameter applies the property twice, to the third and fourth paragraphs in the selection.

Example

The following code example displays the first paragraph of text in the selected text frame, using the **Paragraphs** property:

```
Sub TextCharacters()
With ActiveShape.Text
    MsgBox .Paragraphs(1, 1)
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Paragraphs')} **Related Topics**

A path is the basic component from which objects are constructed. Paths can be open (e.g., a line or curve) or closed (e.g., a circle or polygon). They can also constitute a single line or curve segment or many joined segments.

Artistic text is text type created using the Text tool. Use Artistic text to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. An Artistic text object can contain up to 32,000 characters. CorelDRAW automatically applies the default Artistic text style, which you can change using the Styles Manager.

Text.FitToPath

Sub **FitToPath**(ByRef **pPath** AS **Shape**)

Text

Description

The **FitToPath** method positions artistic text along a defined path in CorelDRAW. You can position artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

When you fit text to a path, you can specify the orientation of characters relative to the path. This allows you to create the impression that letters are standing upright. You can also rotate individual characters to follow the contours of the path. Fitting text to a path also allows you to adjust the spacing between the text and path, specify the vertical position and orientation of the text, specify the horizontal position of text along the path, as well as position text to the opposite side of the path.

CorelDRAW treats text fitted to a path as one object. You can also separate the text from the object. When you separate a text from a curved or closed path, the text retains the shape of the object to which it was fitted. Straightening reverts text to its original appearance.

It is important to note that you can't fit text to the path of another text object.

In order to use the **FitToPath** method, you must pass the path as a parameter:

Parameters	Description
pPath	The **Path** parameter specifies the shape object that will fit text along its path with the **FitToPath** method. A Path is the basic component from which objects are constructed. Paths can be open (e.g., a line or curve) or closed (e.g., a circle or polygon). An open path is a line or curve whose start point and end point are not connected. If you apply a fill to an open path, it is not visible unless the path is closed. A closed is a path that encloses an area because the path's start and end points are connected.

{button ,AL(`CLS_Text;FNC_FitToPath')} **Related Topics**

Text.Find

Function **Find**(ByVal **Text** AS String, ByVal **CaseSensitive** AS Boolean, ByVal StartIndex AS Long, ByVal WrapAround AS Boolean, ByVal IndexingType AS **cdrTextIndexingType**) AS Long

Text

Description

The **Find** method searches for a specified text string in a text range in CorelDRAW. In order to search and locate specific text with the **Find** method, you must pass the **Text**, **CaseSensitive**, **StartIndex**, **WrapAround** and **IndexingType** values as parameters:

Parameters	Description
Text	The **Text** parameter is a text string, contained in quotation marks, indicating the text to search for with the **Find** method. For example, "CorelDRAW Help" is a **Text** parameter.
CaseSensitive	The **CaseSensitive** parameter is a True or False value that indicates if the search string should look for case sensitive text. If the parameter is set to True, the text in the **Text** parameter must match the case of text in the text range.
StartIndex	The **StartIndex** parameter defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame. The default value is 1, which starts with the first text object in a text range.
WrapAround	The **WrapAround** parameter is a True or False value that indicates if text is wrapped around an object. Text is wrapped if it follows the path of an object's shape or bounding box. The default value is False.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example uses the **Find** method to locate the text string "CorelDRAW Help" in the active shape of CorelDRAW. The search string is not case sensitive, and it searches by each word, starting with the first word in the text range. The search string is not wrapped text. If the text string is found, it is removed and replaced with "DRAW" using the **Replace** method. All text matching the old text string is replaced, regardless of case, starting with the first word in the text range:

```
Sub TextFind()
With ActiveShape.Text
    .Find "CorelDRAW Help", False, 1, False, cdrWordIndexing
    .Replace "CorelDRAW Help", "DRAW", False, 1, True, False, cdrWordIndexing
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Find')} **Related Topics**

Text.Replace

Sub **Replace**(ByVal **OldText** AS String, ByVal **NewText** AS String, ByVal **CaseSensitive** AS Boolean, ByVal StartIndex AS Long, ByVal ReplaceAll AS Boolean, ByVal WrapAround AS Boolean, ByVal IndexingType AS **cdrTextIndexingType**)

Text

Description

The **Replace** method searches for a specified text string in a text range in CorelDRAW and replaces the text with a new text string. In order to search and locate specific text with the **Find** method, you must pass the **OldText**, **NewText, CaseSensitive**, **StartIndex**, **ReplaceAll, WrapAround** and **IndexingType** values as parameters:

Parameters	Description
OldText	**OldText** is a text string in quotation marks, indicating the text in a text range that is replaced.
NewText	**NewText** is a text string in quotation marks, indicating the text that is added to a text range, in place of the text specified in the **OldText** parameter.
CaseSensitive	The **CaseSensitive** parameter is a True or False value that indicates if the search string should look for case sensitive text.
StartIndex	The **StartIndex** parameter defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame. The default value is 1
ReplaceAll	The **ReplaceAll** parameter is a True or False value that indicates if all instances of old text found in the text range are replaced with new text. The default value is False.
WrapAround	The **WrapAround** parameter is a True or False value that indicates if text is wrapped around an object. Text is wrapped if it follows the path of an object's shape or bounding box. The default value is False.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0).

Example

The following code example uses the **Replace** method to search the text in the active shape and replace the words "CorelDRAW Help" with "DRAW". All text matching the old text string is replaced, regardless of case, starting with the first word in the text range:

```
Sub TextFind()
With ActiveShape.Text
.Replace "CorelDRAW Help", "DRAW", False, 1, True, False, cdrWordIndexing
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_Replace')} **Related Topics**

Text.LinkToFrame

Sub **LinkToFrame**(ByRef **pVal** AS **Text**)

Text

Description

The **LinkToFrame** method links a text string to a text frame in CorelDRAW. A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.

Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink or enlarge a linked frame, or change the size of the text, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

Parameters	Description
pVal	Description of 'pVal' goes here (**in**)

Example

Example of usage goes here

{button ,AL(`CLS_Text;FNC_LinkToFrame')} **Related Topics**

Text.ImportFromFile

Sub **ImportFromFile**(ByVal **FileName** AS String, ByVal **StartIndex** AS Long, ByVal IndexingType AS **cdrTextIndexingType**)

Text

Description

The **ImportFromFile** method imports text from a specified file and places the text into CorelDRAW. In order to import text with the **ImportFromFile** method, you must pass the **FileName**, **StartIndex**, and **IndexingType** values as parameters:

Parameters	Description
FileName	**FileName** is the full path name of a file containing text that will be imported into CorelDRAW. The path name is the location of a folder or file on a computer. For example, Corel application files are stored in the path C:\COREL\ by default. This means that the files are stored in a folder called COREL on the C: drive.
StartIndex	The **StartIndex** parameter defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame. The default value is 1.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0)

Example

The following code example imports all the text located in a specified file into the active shape of CorelDRAW starting with the second word in the file, using the **ImportFromFile** method. The file is a text file named "Import and Export" and is stored on the root of the local C drive:

```
Sub TextImport()
With ActiveShape.Text
    .ImportFromFile "c:/Import and Export.txt", 2, cdrWordIndexing
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_ImportFromFile')} **Related Topics**

Text.ExportToFile

Sub **ExportToFile**(ByVal **FileName** AS String, ByVal **StartIndex** AS Long, ByVal **Count** AS Long, ByVal IndexingType AS **cdrTextIndexingType**)

Text

Description

The **ExportToFile** method exports a text range from CorelDRAW to a separate file. In order to export text with the **ExportToFile** method, you must pass the **FileName**, **StartIndex**, **Count** and **IndexingType** values as parameters:

Parameters	Description
FileName	**FileName** is the full path name of a file containing text that will be imported into CorelDRAW. The path name is the location of a folder or file on a computer. For example, Corel application files are stored in the path C:\COREL\ by default. This means that the files are stored in a folder called COREL on the C: drive.
StartIndex	The **StartIndex** parameter defines the first text object in a text range. If the index type is Word, and the start index is 3, the **StartIndex** parameter begins the range at the third word in the selected text frame. The default value is 1.
Count	**Count** defines how much text is exported within a text range. If the start index is 3, count is 2, and indexing type is by word, the **ExportToFile** method exports 2 words, the third and fourth to a specified file.
IndexingType	**IndexingType** defines what portion of text sets a range. The indexing type can be character, word, or paragraph and the **IndexingType** parameter returns a value of cdrIndexingType. The default value is cdrCharacterIndexing (0)

Example

The following code example exports text in the active shape of CorelDRAW into a specified file, using the **ExportToFile** method. Only the second and third words in the shape's text range are exported. The file is a text file named "Import and Export" and is stored on the root of the local C drive:

```
Sub TextExport()
With ActiveShape.Text
    .ExportToFile "c"/Import and Export.txt", 2, 2, cdrWordIndexing
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_ExportToFile')} **Related Topics**

Text.ConvertToArtistic

Sub **ConvertToArtistic**()

Text

Description

The **ConvertToArtistic** method changes paragraph text to artistic text in CorelDRAW. You create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

Example

The following code example converts the paragraph text in the active shape to artistic text, using the **ConvertToArtistic** method:

```
Sub TextConvertArtistic()
With ActiveShape.Text
    .ConvertToArtistic
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_ConvertToArtistic')} **Related Topics**

Text.ConvertToParagraph

Sub **ConvertToParagraph**()

Text

Description

The **ConvertToParagraph** method changes artistic text to paragraph text in CorelDRAW. You create both Paragraph text and Artistic text with the Text tool. Artistic text is used to add single lines of text, such as titles, or to apply graphic effects, such as fitting text to a path, creating extrusions and blends, and creating all other special effects. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects.

Example

The following code example converts the artistic text in the active shape to paragraph text, using the **ConvertToArtistic** method. The paragraph text is made HTML compatible using the **MakeHTMLCompatible** method:

```
Sub TextConvertParagraph()
With ActiveShape.Text
    .ConvertToParagraph
    .MakeHTMLCompatible
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_ConvertToParagraph')} **Related Topics**

Text.IsHTMLCompatible

Property **IsHTMLCompatible** AS Boolean

Text

Description

The **IsHTMLCompatible** property returns a True or False value indicating the HTML compatibility of text in CorelDRAW. Hypertext Markup Language (HTML) is the World Wide Web authoring standard. HTML is comprised of markup tags that define the structure and components of a document. The tags are used to tag text and integrate resources (such as images, sound, video, and animation) when you create a Web page. If the property is set to true, the text is HTML compatible.

You can convert standard Paragraph text to HTML text so that you can edit the text of your published document in a Web browser. If you don't convert Paragraph text to HTML text before you publish your document to the Internet, the text is converted to a bitmap when published and cannot be edited in a browser. Artistic text cannot be converted to HTML text and is always treated as a bitmap.

Like other Internet objects in your document, HTML text resides on the Internet layer. The Internet layer is created automatically when you convert your text to HTML.

The **IsHTMLCompatible** returns a Read-Only value.

Example

The following code example displays the HTML compatibility status of the active shape's text in a message box:

```
Sub TextHTML()
With ActiveShape.Text
    MsgBox .IsHTMLCompatible
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_IsHTMLCompatible')} **Related Topics**

Text.MakeHTMLCompatible

Function **MakeHTMLCompatible**(ByVal **HTML** AS Boolean) AS Boolean

Text

Description

The **MakeHTMLCompatible** property makes paragraph text HTML compatible in CorelDRAW. Hypertext Markup Language (HTML) is the World Wide Web authoring standard. HTML is comprised of markup tags that define the structure and components of a document. The tags are used to tag text and integrate resources (such as images, sound, video, and animation) when you create a Web page.

You can convert standard Paragraph text to HTML text so that you can edit the text of your published document in a Web browser. If you don't convert Paragraph text to HTML text before you publish your document to the Internet, the text is converted to a bitmap when published and cannot be edited in a browser. Artistic text cannot be converted to HTML text and is always treated as a bitmap.

Like other Internet objects in your document, HTML text resides on the Internet layer. The Internet layer is created automatically when you convert your text to HTML.

Parameters	Description
HTML	HTML is a True or False value indicating the HTML compatibility of text. Hypertext Markup Language (HTML) is the World Wide Web authoring standard. HTML is comprised of markup tags that define the structure and components of a document. The tags are used to tag text and integrate resources (such as images, sound, video, and animation) when you create a Web page. If the value is True, the text is HTML compatible.

Example

The following code example converts the artistic text in the active shape to paragraph text, using the **ConvertToArtistic** method. The paragraph text is made HTML compatible using the **MakeHTMLCompatible** method:

```
Sub TextMakeHTML()
With ActiveShape.Text
    .ConvertToParagraph
    .MakeHTMLCompatible
End With
End Sub
```

{button ,AL(`CLS_Text;FNC_MakeHTMLCompatible')} **Related Topics**

Connector properties

Legend

EndPoint
StartPoint

Connector

Class **Connector**

Properties Referenced by

The **Connector** class defines the characteristics of connector line objects and describes the look and behavior of the objects through its properties and methods.

A connector line object is a straight line that acts as a dynamic join between two objects in a drawing, created with either the Connector Line tool or the Interactive Connector tool. If you move one or both of the objects, the connector line (also called a flow line) is adjusted accordingly. If only one end of a line is connected to an object, the other end is fixed to the page. You can only reposition a connector line by moving the objects to which it is attached.

For connector lines to be effective they must be linked to the objects they are labeling. To link objects, use the object's snap points.

You can change the outline of connector lines as well as customize connector line styles and arrowheads.

{button ,AL(`CLS_Connector')} **Related Topics**

Connector.StartPoint

Property **StartPoint** AS **ShapePoint**

Connector

Description

The **StartPoint** property returns or sets the starting point of the connector line object in CorelDRAW. This starting point will be a specified object's snap point. When you using use connector lines, special points on each object, called snap points, are activated. When a mouse passes over a snap point, it becomes visible. Objects can only be linked by connector lines at snap points, which are generally located at each node and at the center of the object.

`StartPoint(5)` will begin the connector line at the fifth snap point of the object.

Example

The following code example sets the reference point of the document to the lower left corner and creates a new shape point with coordinates (1, 6), one inch above and 6 inches to the right of the reference point. A rectangle and ellipse are created, and a connector line is drawn between them, starting at the tenth snap point of the rectangle to the first snap point of the ellipse:

```
Sub PointStart()
Dim sp as New ShapePoint
Dim s1 As Shape, s2 As Shape, c As Shape
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
Set s1 = ActiveLayer.CreateRectangle(0, 0, 3, 3)
Set s2 = ActiveLayer.CreateEllipse(4, 4, 6, 1)
Set c = ActiveLayer.CreateConnector(s1.Points(10), s2.Points(1))
    c.Connector.StartPoint = s1.Points(10)
    c.Connector.EndPoint = s2.Points(1)
End Sub
```

{button ,AL(`CLS_Connector;FNC_StartPoint')} **Related Topics**

When you using use connector lines, special point on each object, called snap points, are activated. When a mouse passes over a snap point, it becomes visible. Objects can only be linked by connector lines at snap points, which are generally located at each node and at the center of the object.

Connector.EndPoint

Property **EndPoint** AS **ShapePoint**

Connector

Description

The **EndPoint** property returns or sets the ending point of the connector line object in CorelDRAW. This ending point will be a specified object's snap point. When you using use connector lines, special points on each object, called snap points, are activated. When a mouse passes over a snap point, it becomes visible. Objects can only be linked by connector lines at snap points, which are generally located at each node and at the center of the object.

`EndPoint(5)` will begin the connector line at the fifth snap point of the object.

Example

The following code example sets the reference point of the document to the lower left corner and creates a new shape point with coordinates (1, 6), one inch above and 6 inches to the right of the reference point. A rectangle and ellipse are created, and a connector line is drawn between them, starting at the tenth snap point of the rectangle to the first snap point of the ellipse:

```
Sub PointStart()
Dim sp as New ShapePoint
Dim s1 As Shape, s2 As Shape, c As Shape
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
Set s1 = ActiveLayer.CreateRectangle(0, 0, 3, 3)
Set s2 = ActiveLayer.CreateEllipse(4, 4, 6, 1)
Set c = ActiveLayer.CreateConnector(s1.Points(10), s2.Points(1))
    c.Connector.StartPoint = s1.Points(10)
    c.Connector.EndPoint = s2.Points(1)
End Sub
```

{button ,AL(`CLS_Connector;FNC_EndPoint')} **Related Topics**

Guide properties

Guide Legend

- Preset

- Type

Guide methods

Guide Legend

MakeEditable

Guide

Class **Guide**

Properties Methods Referenced by

The Guide class defines the characteristics of guideline objects and describes the look and behavior of the objects through its properties and methods. A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guidelines may be Horizontal, Vertical, or Slanted and any number of them may be created and saved with your drawing. Guidelines return a value of cdrGuideType. You can also enable snapping guidelines so that objects automatically align with the guidelines when moved or drawn nearby.

Guidelines may be selected, rotated, nudged, duplicated, and deleted. Selected guidelines always appear red, but guidelines may be made any color available in the CorelDRAW palette.

By default, guidelines do not appear on the printed page but they can be printed by changing the appropriate settings in the Object Manager.

Preset guidelines

Preset guidelines make it easy to apply frequently used guidelines to your drawings. You can create your own presets, and you can apply those presets to all new documents. If you resize or reorient the page after applying a preset, the guidelines regenerate in the appropriate places. Repositioning a preset guideline changes it to a standard guideline.

Preset guidelines are available only on default pages in multi-page documents, and not on individually resized or oriented pages. Preset guidelines must be deleted in the Options dialog box - they cannot be selected and deleted on the drawing page. If you delete a preset any other way, it regenerates when you resize or reorient the page.

Standard guidelines

You set up Horizontal and Vertical guidelines based on the horizontal or vertical distance from the 0 point on the appropriate ruler. Conversely, you set up Slanted guidelines based on two specific ruler coordinates or one coordinate and an angle.

For more speed than precision, use the mouse to add guidelines to your drawing. You can create Horizontal and Vertical guidelines by dragging from a ruler to the Drawing Window. To create Slanted guidelines, rotate a Horizontal or Vertical guideline.

Any guidelines you add appear on every page of a multi-page document except for individually resized or reoriented pages. Standard guidelines can be deleted on the drawing page by selecting them with the Pick tool and deleting them. If you delete a guideline in a multi-page document, the guideline is removed from all pages.

{button ,AL(`CLS_Guide')} **Related Topics**

A guideline is a line placed anywhere in the CorelDRAW drawing window that is used to help align and position drawing objects. Guidelines are used with the rulers and the grid to draw objects with precision.

Guide.Type

Property **Type** AS **cdrGuideType**

Guide

Description

The **Type** property returns the specific kind of guideline used in CorelDRAW. A guideline may be one of three types:

Horizontal guideline - Based on the horizontal distance from the 0 point on the appropriate ruler, this guideline appears left to right on the drawing page

Vertical guideline - Based on the vertical distance from the 0 point on the appropriate ruler, this guideline appears top to bottom on the drawing page

Slanted guideline - Based on two specific ruler coordinates or one coordinate and an angle, this guideline appears diagonally across the drawing page.

The **Type** property returns a Read-Only value.

Example

The following code example looks at each guideline in the master document. Every guideline that is vertical is counted as 1 vertical guideline. Every guideline that is horizontal is counted as 1 horizontal guideline. Every guideline that is slanted is counted as 1 slanted guideline. A message box displays the number of horizontal, vertical, and slanted guidelines that are in the master document:

```
Sub GuideType()
Dim MyShape As Shape
Dim VerticalGuide As Long, HorizontalGuide As Long, SlantedGuide As Long
For Each MyShape In ActiveDocument.Pages(0).Guides(cdrAllGuides)
'looks at each guideline in the master document
'0 denotes the active page
    Select Case MyShape.Guide.Type
    Case cdrVerticalGuide
      VerticalGuide = VerticalGuide + 1
      'each instance of a vertical guideline is counted as 1 vertical guideline
    Case cdrHorizontalGuide
      HorizontalGuide = HorzontalGuide + 1
      'each instance of a horizontal guideline is counted as 1 horizontal guideline
    Case cdrSlantedGuide
      SlantedGuide = SlantedGuide + 1
      'each instance of a slanted guideline is counted as 1 slanted guideline
    End Select
Next MyShape
MsgBox "The master document contains: " & vbCrLf _
& HorizontalGuide & " horizontal guidelines" & vbCrLf _
& VerticalGuide & " vertical guidelines" & vbCrLf _
& SlantedGuide & " slanted guidelines"
End Sub
```

{button ,AL(`CLS_Guide;FNC_Type')} **Related Topics**

Guide.Preset

Property **Preset** AS Boolean

Guide

Description

The Preset property returns a True or False value indicating if a guideline is a preset guideline. Preset guidelines make it easy to apply frequently used guidelines to your drawings. You can create your own presets, and you can apply those presets to all new documents. If you resize or reorient the page after applying a preset, the guidelines regenerate in the appropriate places.

Preset guidelines are available only on default pages in multi-page documents, and not on individually resized or oriented pages. Repositioning a preset guideline changes it to a standard guideline.

If a guideline is a preset guideline, the Preset property will return a value of True. If it is not a preset guideline, a value of False is returned.

The Preset property returns a Read-Only value.

Example

The following code example looks and each guideline in the master document and deletes all guidelines that are preset guidelines:

```
Sub PresetRemove()
Dim MyShape As Shape
For Each MyShape In ActiveDocument.Pages(0).Guides(cdrAllGuides)
'looks at each guideline in the master document
'0 denotes the master page
    If MyShape.Guide.Preset Then
    MyShape.Delete
    'deletes all guidelines that are preset guidelines
    End If
Next MyShape
End Sub
```

{button ,AL(`CLS_Guide;FNC_Preset')} **Related Topics**

Guide.MakeEditable

Sub **MakeEditable**()

Guide

Description

The MakeEditable method unlocks a preset guideline so that changes or adjustments can be made to the guideline properties. Locking guidelines prevents it from being selected, moved or deleted accidentally. By applying the MakeEditable method to a preset guideline, you are changing the guideline to a standard guideline that may be edited directly on the drawing page.

Example

The following code example looks at all the guidelines in the master document and uses the MakeEditable method to convert all preset guidelines in the active document to standard guidelines:

```
Sub EditGuideline()
Dim MyShape As Shape
For Each MyShape In ActiveDocument.Pages(0).Guides(cdrAllGuides)
'looks at each guideline in the master document
'0 denotes the master page
    If MyShape.Guide.Preset Then
    MyShape.Guide.MakeEditable
    End If
Next MyShape
End Sub
```

{button ,AL(`CLS_Guide;FNC_MakeEditable')} **Related Topics**

ShapePoint properties

ShapePoint Legend

- Application
- Node
- Parent
  - PositionX
  - PositionY
- Type

ShapePoint

Class **ShapePoint**

Properties Referenced by

The **ShapePoint** class defines the characteristics of **ShapePoint** objects and describes the look and behavior of the objects through its properties and methods.

A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

The shape point type returns a value of cdrPointType.

{button ,AL(`CLS_ShapePoint')} **Related Topics**

ShapePoint.Application

Property **Application** AS **Application**

ShapePoint

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example sets the reference point of the page to the bottom left corner and creates a new shape point object with the coordinates (1,6). The latest version of the application, the shape point type, and the X and Y coordinates of the point display in a message box:

```
Sub PointStart()
Dim sp as New ShapePoint
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
With sp
    .MsgBox .Application.Version
    .MsgBox .Type
    .MsgBox .PositionX
    .MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_Application')} **Related Topics**

ShapePoint.Parent

Property **Parent** AS **Shape**

ShapePoint

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the **Points** collection's first point's parent layer in a message box:

```
Sub LayerParent()
With ActiveShape.Points (1)
    MsgBox .Parent.Layer.Name
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_Parent')} **Related Topics**

ShapePoint.PositionX

Property **PositionX** AS Double

ShapePoint

Definition

The **PositionX** property returns or sets the horizontal position of a shape point in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

Example

The following code example sets the reference point of the page to the bottom left corner and creates a new shape point object with the coordinates (1,6). The latest version of the application, the shape point type, and the X and Y coordinates of the point display in a message box:

```
Sub PointStart()
Dim sp as New ShapePoint
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
With sp
    .MsgBox .Application.Version
    .MsgBox .Type
    .MsgBox .PositionX
    .MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_PositionX')} **Related Topics**

ShapePoint.PositionY

Property **PositionY** AS Double

ShapePoint

Definition

The **PositionY** property returns or sets the vertical position of a shape point in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

Example

The following code example sets the reference point of the page to the bottom left corner and creates a new shape point object with the coordinates (1,6). The latest version of the application, the shape point type, and the X and Y coordinates of the point display in a message box:

```
Sub PointStart()
Dim sp as New ShapePoint
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
With sp
    .MsgBox .Application.Version
    .MsgBox .Type
    .MsgBox .PositionX
    .MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_PositionY')} **Related Topics**

ShapePoint.Type

Property **Type** AS **cdrPointType**

ShapePoint

Description

The **Type** property returns the shape point type of a point object in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

The **Type** property returns a Read-Only value. The **Type** property returns cdrPointType.

Example

The following code example sets the reference point of the page to the bottom left corner and creates a new shape point object with the coordinates (1,6). The latest version of the application, the shape point type, and the X and Y coordinates of the point display in a message box:

```
Sub PointStart()
Dim sp as New ShapePoint
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
With sp
    .MsgBox .Application.Version
    .MsgBox .Type
    .MsgBox .PositionX
    .MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_Type')} **Related Topics**

cdrShapePoint=0

cdrFreePoint=1

ShapePoint.Node

Property **Node** AS **Node**

ShapePoint

Description

The **Node** property returns the node associated with a shape point object in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

A node is a small square on a line, curve, or object outline used to edit the object. Curve objects may have numerous nodes, created along the curve's subpaths and each node is identified by its index number.

The **Node** property returns a Read-Only value.

Example

The following code example sets the reference point of the page to the bottom left corner and creates a new shape point object with the coordinates (1,6). The node type displays in a message box:

```
Sub PointStart()
Dim sp as New ShapePoint
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
With sp
    .MsgBox .Node.Type
End With
End Sub
```

{button ,AL(`CLS_ShapePoint;FNC_Node')} **Related Topics**

Points properties

Points Legend

- Application
- Count
-
- Item
- Parent

Points

Class **Points**

Properties Referenced by

The **Points** class defines the characteristics of **ShapePoint** collection objects and describes the look and behavior of the objects through its properties and methods.

A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

The shape point type returns a value of cdrPointType.

{button ,AL(`CLS_Points')} **Related Topics**

Points.Application

Property **Application** AS **Application**

Points

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PointsApp()
With ActiveShape.Points.Item(2)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Points;FNC_Application')} **Related Topics**

Points.Parent

Property **Parent** AS **Shape**

Points

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the **Points** collection's parent layer in a message box:

```
Sub LayerParent()
With ActiveShape.Points
    MsgBox .Parent.Layer.Name
End With
End Sub
```

{button ,AL(`CLS_Points;FNC_Parent')} **Related Topics**

Points.Item

Property **Item**(ByVal **Index** AS Long) AS **ShapePoint**

Points

Description

The **Item** property returns a value associated with the index number of shape point object in the **Points** collection of CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

Each shape point in CorelDRAW is identified by an index number. Points(5) refers to the fifth point in the collection. The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Points.Item(5)` is the same as `Points(5)` - they both reference the fifth point in the collection.

You must reference a point in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Points** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PointsApp()
With ActiveShape.Points.Item(2)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Points;FNC_Item')} **Related Topics**

Points.Count

Property **Count** AS Long

Points

Description

The **Count** property returns the number of shape point objects in the **Points** collection of CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of shape point objects in the **Points** collection of the active shape:

```
Sub PointsCollection()
With ActiveShape.Points
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Points;FNC_Count')} **Related Topics**

CrossPoint properties

CrossPoint Legend

- Offset
- PositionX
- PositionY

CrossPoint

Class **CrossPoint**

Properties Referenced by

The **CrossPoint** class defines the characteristics of CrossPoint objects and describes the look and behavior of the objects through its properties and methods. A CrossPoint object represents an intersecting point on a shape segment. A crosspoint is created when two or more intersecting shape object's are integrated into a selection in CorelDRAW.

A crosspoint is defined by an x and y coordinate representing the horizontal and vertical placement of the intersection point.

{button ,AL(`CLS_CrossPoint')} **Related Topics**

CrossPoint.PositionX

Property **PositionX** AS Double

CrossPoint

Definition

The **PositionX** property returns the horizontal position of a shape object's crosspoint intersection in CorelDRAW. A crosspoint is an intersection created when two or more shapes are integrated in a selection and it is defined by its x (horizontal) and y (vertical) coordinates.

The **PositionX** property returns a Read-Only value.

Example

The following code example creates two separate ellipses with the CreateEllipse method. The active document is cleared, both ellipses are added to the active selection, and the ellipses are combined to create a single shape object. The intersections, or CrossPoints, of this new shape are identified by creating small circles around each point, using its x and y coordinates to serve as the coordinates of the small circles:

```
Sub XPosition()
Dim s1 As Shape, s2 As Shape
Dim s As Shape
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s1 = ActiveLayer.CreateEllipse(4, 2, 1, 0)
Set s2 = ActiveLayer.CreateEllipse(3, 1, 2, 5)
    ActiveDocument.ClearSelection
    s1.AddToSelection
    s2.AddToSelection
    ActiveSelection.Combine
Set s = ActiveSelection.Shapes(1)
Set cps = s.Curve.Subpaths(1).GetIntersections(s.Curve.Subpaths(2))
For Each cp In cps
    ActiveLayer.CreateEllipse2 cp.PositionX, cp.PositionY, 0.1
Next cp
End Sub
```

{button ,AL(`CLS_CrossPoint;FNC_PositionX')} **Related Topics**

CrossPoint.PositionY

Property **PositionY** AS Double

CrossPoint

Description

The PositionY property returns the vertical position of a shape object's CrossPoint intersection in CorelDRAW. A CrossPoint is an intersection created when two or more shapes are integrated in a selection and it is defined by its x (horizontal) and y (vertical) coordinates.

The PositionY property returns a Read-Only value.

Example

The following code example creates two separate ellipses with the CreateEllipse method. The active document is cleared, both ellipses are added to the active selection, and the ellipses are combined to create a single shape object. The intersections, or CrossPoints, of this new shape are identified by creating small circles around each point, using its x and y coordinates to serve as the coordinates of the small circles:

```
Sub YPosition()
Dim s1 As Shape, s2 As Shape
Dim s As Shape
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s1 = ActiveLayer.CreateEllipse(4, 2, 1, 0)
Set s2 = ActiveLayer.CreateEllipse(3, 1, 2, 5)
    ActiveDocument.ClearSelection
    s1.AddToSelection
    s2.AddToSelection
    ActiveSelection.Combine
Set s = ActiveSelection.Shapes(1)
Set cps = s.Curve.Subpaths(1).GetIntersections(s.Curve.Subpaths(2))
For Each cp In cps
    ActiveLayer.CreateEllipse2 cp.PositionX, cp.PositionY, 0.1
Next cp
End Sub
```

{button ,AL(`CLS_CrossPoint;FNC_PositionY')} **Related Topics**

CrossPoint.Offset

Property **Offset** AS Double

CrossPoint

Description

The Offset property returns the offset distance, in document units, that an object's first node is moved out of line from another object in CorelDRAW. A duplicate object (clone) is offset from its original object (parent) when it is created - it is moved a short distance from the parent object. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. The default value of the offset distance in CorelDRAW is .5 document units.

The Offset property returns a Read-Only value.

Example

The following code example creates a new line in the active document with the CreateLineSegment method and creates a new subpath of the line with the CreateSubPath method. The subpath is not appended to the original line, leaving the lines to be treated as two separate objects. The GetIntersections method determines the intersection point where to two lines meet and the lines are broken at that crosspoint. That crosspoint now serves as the first node of the detached line segment that is offset, or moved slightly, from the other line segment:

```
Sub CrossPointOffset()
Dim s As Shape
Dim spath As SubPath
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s = ActiveLayer.CreateLineSegment(0, 0, 4, 4)
Set spath = s.Curve.CreateSubPath(0, 4)
    spath.AppendLineSegment False, 4, 0
Set cps = s.Curve.Segments(1).GetIntersections(s.Curve.Segments(2))
    s.Curve.Subpaths(1).BreakApartAt cps(1).Offset
End Sub
```

{button ,AL(`CLS_CrossPoint;FNC_Offset')} **Related Topics**

Offset refers to a movement out of line with another object. An offset object is distanced from another specified object and this distance is the offset value.

CrossPoints properties

CrossPoints Legend

- Count

- Item

CrossPoints

Class **CrossPoints**

Properties Referenced by

The **CrossPoints** class defines the characteristics of CrossPoint collection objects and describes the look and behavior of the objects through its properties and methods. A CrossPoint object represents an intersecting point on a shape segment. A crosspoint is created when two or more intersecting shape object's are integrated into a selection in CorelDRAW.

A crosspoint is defined by an x and y coordinate representing the horizontal and vertical placement of the intersection point.

{button ,AL(`CLS_CrossPoints')} **Related Topics**

CrossPoints.Item

Property **Item**(ByVal **Index** AS Long) AS **CrossPoint**

CrossPoints

Description

The **Item** property returns a string value associated with the index number of a crosspoint in the **DataItems** collection of CorelDRAW. A CrossPoint object represents an intersecting point on a shape segment. A crosspoint is created when two or more intersecting shape object's are integrated into a selection in CorelDRAW.

A crosspoint is defined by an x and y coordinate representing the horizontal and vertical placement of the intersection point.

`CrossPoints(2)` refers to the second crosspoint object in the **CrossPoints** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `CrossPoints.Item(2)` is the same as `CrossPoint(2)` - they both reference the second crosspoint object in the **CrossPoints** collection.

You must reference a crosspoint object in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **CrossPoints** collection; it uniquely identifies each member of the collection.

Example

The following code example creates two separate ellipses with the **CreateEllipse** method. The active document is cleared, both ellipses are added to the active selection, and the ellipses are combined to create a single shape object. The intersections, or CrossPoints, of this new shape are identified by creating small circles around each point, using its x and y coordinates to serve as the coordinates of the small circles. The number of crosspoints and the X position of the first crosspoint display in a message box:

```
Sub XPosition()
Dim s1 As Shape, s2 As Shape
Dim s As Shape
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s1 = ActiveLayer.CreateEllipse(4, 2, 1, 0)
Set s2 = ActiveLayer.CreateEllipse(3, 1, 2, 5)
    ActiveDocument.ClearSelection
    s1.AddToSelection
    s2.AddToSelection
    ActiveSelection.Combine
Set s = ActiveSelection.Shapes(1)
Set cps = s.Curve.Subpaths(1).GetIntersections(s.Curve.Subpaths(2))
MsgBox cps.Count
MsgBox cps.Item(1).PositionX
End Sub
```

{button ,AL(`CLS_CrossPoints;FNC_Item')} **Related Topics**

CrossPoints.Count

Property **Count** AS Long

CrossPoints

Description

The **Count** property returns the number of crosspoints in the **CrossPoints** collection of CorelDRAW. A CrossPoint object represents an intersecting point on a shape segment. A crosspoint is created when two or more intersecting shape object's are integrated into a selection in CorelDRAW.

A crosspoint is defined by an x and y coordinate representing the horizontal and vertical placement of the intersection point.

This property returns a Read-Only value.

Example

The following code example creates two separate ellipses with the **CreateEllipse** method. The active document is cleared, both ellipses are added to the active selection, and the ellipses are combined to create a single shape object. The intersections, or CrossPoints, of this new shape are identified by creating small circles around each point, using its x and y coordinates to serve as the coordinates of the small circles. The number of crosspoints and the X position of the first crosspoint display in a message box:

```
Sub XPosition()
Dim s1 As Shape, s2 As Shape
Dim s As Shape
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s1 = ActiveLayer.CreateEllipse(4, 2, 1, 0)
Set s2 = ActiveLayer.CreateEllipse(3, 1, 2, 5)
    ActiveDocument.ClearSelection
    s1.AddToSelection
    s2.AddToSelection
    ActiveSelection.Combine
Set s = ActiveSelection.Shapes(1)
Set cps = s.Curve.Subpaths(1).GetIntersections(s.Curve.Subpaths(2))
MsgBox cps.Count
MsgBox cps.Item(1).PositionX
End Sub
```

{button ,AL(`CLS_CrossPoints;FNC_Count')} **Related Topics**

RecentFile properties

RecentFile Legend

- Application
  - FullName
- Index
  - Name
- Parent
  - Path

RecentFile methods

RecentFile Legend

- Delete
- Open

RecentFile

Class **RecentFile**

Properties Methods Referenced by

The **RecentFile** class defines the characteristics of recent files objects and describes the look and behavior of the objects through its properties and methods.

A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

CorelDRAW keeps the four most recently opened documents in the Recent Files list.

{button ,AL(`CLS_RecentFile')} **Related Topics**

RecentFile.Delete

Sub **Delete**()

RecentFile

Description

The **Delete** method removes a specified document from the Recent Files list in CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The document to be removed is identified by its placeholder in the Recent Files list. For example, `RecentFiles(1).Delete` removes the first document listed in the Recent Files list.

It is important to note that removing a document with the **Delete** method does not delete the entire document - it only removes its name from the Recent Files list in CorelDRAW.

Example

The following code example removes the third document from the Recent Files list:

```
Sub FileRemove()
RecentFiles.Item(3).Delete
'since Item is the default property, it does not have to be included
'RecentFiles(3).Delete will also remove the file
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Delete')} **Related Topics**

RecentFile.Open

Function **Open**() AS Object

RecentFile

Description

The **Open** method opens a document for editing from the Recent Files list in CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The document to be opened is identified by its placeholder in the Recent Files list. For example, `RecentFiles(1).Open` opens the first document listed in the Recent Files list.

It is important to note that opening a document removing a document with the **Delete** method does not delete the entire document. It only removes its name from the Recent Files list in CorelDRAW.

Example

The following code opens the second document listed in the Recent Files list of CorelDRAW:

```
Sub FileOpen()
RecentFiles(2).Open
'since Item is the default property, it does not have to be included
'RecentFiles(2).Open will also open the file
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Open')} **Related Topics**

RecentFile.Application

Property **Application** AS Object

RecentFile

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub FileApp()
With RecentFiles(1).Application
    MsgBox .Version
End With
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Application')} **Related Topics**

RecentFile.Parent

Property **Parent** AS **RecentFiles**

RecentFile

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the parent path of the first document in the Recent Files list in a message box:

```
Sub FileParent()
With RecentFiles(1).Parent
    MsgBox .Parent.Path
End With
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Parent')} **Related Topics**

RecentFile.Index

Property **Index** AS Long

RecentFile

Description

The **Index** property returns a value associated with the location of a document in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The index number is a unique placeholder for a document in the list. For example, `RecentFiles(1)` refers to the first document listed in the Recent Files list. A document with an index of 1 holds the first position in the Recent Files list of CorelDRAW.

The **Index** property returns a Read-Only value.

Example

The following code example displays the index number of a document in the Recent Files list of CorelDRAW in a message box:

```
Sub FileIndex()
MsgBox RecentFiles(1).Index
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Index')} **Related Topics**

RecentFile.Name

Property **Name** AS String

RecentFile

Description

The **Name** property returns a string value associated with the identity of a document in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The **Name** property is a Read-Only value.

Example

The following code example displays the name of a the first document in the Recent Files list in a message box:

```
Sub FileName()
MsgBox RecentFiles(1).Name
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Name')} **Related Topics**

RecentFile.Path

Property **Path** AS String

RecentFile

Description

The **Path** property returns a string value associated with the computer location of a document in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The **Path** property is a Read-Only value.

Example

The following code example displays the path of the first document listed in the Recent Files list in a message box:

```
Sub FilePath()
MsgBox RecentFiles(1).Path
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_Path')} **Related Topics**

RecentFile.FullName

Property **FullName** AS String

RecentFile

Description

The **FullName** property returns a string value associated with the document name and computer location of a document in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The **FullName** property is a Read-Only value.

Example

The following code example displays the full name and path of the first document in the Recent Files list in a message box:

```
Sub FileFullName()
MsgBox RecentFiles(1).FullName
End Sub
```

{button ,AL(`CLS_RecentFile;FNC_FullName')} **Related Topics**

RecentFiles properties

RecentFiles Legend

- Application
- Count
-
- Item
- Maximum
- Parent

RecentFiles methods

RecentFiles Legend

Add

RecentFiles

Class **RecentFiles**

Properties Methods Referenced by

The **RecentFiles** class defines the characteristics of recent files collection objects and describes the look and behavior of the collection objects through its properties and methods.

A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

CorelDRAW keeps the four most recent documents opened in the Recent Files list.

{button ,AL(`CLS_RecentFiles')} **Related Topics**

RecentFiles.Add

Function **Add**(ByVal **Name** AS String, ByVal **Path** AS String) AS **RecentFile**

RecentFiles

Description

The **Add** method places a specified document into the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

In order to add a document to the **RecentFiles** collection with the **Add** property, you must pass the Name and Path of the document as parameters:

Parameters	Description
Name	The **Name** property returns a string value associated with the identity of a document in the Recent Files list of CorelDRAW.
Path	The **Path** property returns a string value associated with the computer location of a document in the Recent Files list of CorelDRAW.

Example

The following code example adds the document "My CorelDRAW.cdr" to the Recent Files list of CorelDRAW:

```
Sub FilesAdd()
RecentFiles.Add "My CorelDRAW.cdr", C:
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Add')} **Related Topics**

RecentFiles.Application

Property **Application** AS Object

RecentFiles

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub FilesApp()
With RecentFiles.Application
    MsgBox .Version
End With
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Application')} **Related Topics**

RecentFiles.Parent

Property **Parent** AS Object

RecentFiles

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the parent path of the Recent Files list in a message box:

```
Sub FilesParent()
With RecentFiles.Parent
    MsgBox .Path
End With
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Parent')} **Related Topics**

RecentFiles.Item

Property **Item**(ByVal **Index** AS Long) AS **RecentFile**

RecentFiles

Description

The **Item** property returns a value associated with the location of a document in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The Item property is a unique placeholder for a document in the list. For example, `RecentFiles.Item(1)` refers to the first document listed in the Recent Files list. A document with an Item number of 1 holds the first position in the Recent Files list of CorelDRAW.

The **Item** property is equivalent to the **Index** property of the **RecentFile** class. Both properties reference a document in the Recent Files list of CorelDRAW.

The **Index** property returns a Read-Only value.

In order to reference an item in the **RecentFiles** collection with the **Item** property, you must pass the index number of the document as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **RecentFiles** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the name of the first document in the Recent Files list of CorelDRAW in a message box:

```
Sub FilesItem()
With RecentFiles.Item(1)
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Item')} **Related Topics**

RecentFiles.Count

Property **Count** AS Long

RecentFiles

Description

The **Count** property returns the number of documents in the **RecentFiles** collection of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of documents in the Recent Files list of CorelDRAW in a message box:

```
Sub FilesCount()
MsgBox "There are " & RecentFiles.Count & " documents in the Recent Files list."
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Count')} **Related Topics**

RecentFiles.Maximum

Property **Maximum** AS Long

RecentFiles

Description

The **Maximum** property returns a value associated with the number of documents that can be stored in the Recent Files list of CorelDRAW. A recent file is a document that has been in use in the current session of CorelDRAW. All recent files are listed in the CorelDRAW application window, located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The Maximum property returns a Read-Only value.

Example

The following code example displays the maximum number of documents that can be stored in the Recent Files list of CorelDRAW in a message box:

```
Sub FilesMax()
MsgBox RecentFiles.Maximum
End Sub
```

{button ,AL(`CLS_RecentFiles;FNC_Maximum')} **Related Topics**

Node properties

Node Legend

- AbsoluteIndex
- Application
- Index
- IsEnding
- Parent
  - PositionX
  - PositionY
- Segment
- SubPath
- SubPathIndex
  - Type

Node methods

Node Legend

- BreakApart
- ConnectWith
- Delete
- GetDistanceFrom
- GetPosition
- JoinWith
- Move
- Next
- Previous
- SetPosition

Node

Class **Node**

Properties Methods Referenced by

The **Node** class defines the characteristics of node objects and describes the look and behavior of the objects through its properties and methods.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the type of node (smooth, cusp, or symmetrical) and segment (line or curve) selected.

There are three types of nodes and the control points of each node type behave differently:

Cusp node

The control points of a cusp node move independently from one another. A curve that passes through a cusp node can bend at a sharp angle.

Smooth node

The control points of a smooth node are always directly opposite each other. When you move one control point, the other moves also. The control points of a smooth node may be of different lengths. Smooth nodes produce a smooth transition between line segments.

Symmetrical node

The control points of a symmetrical node are always directly opposite each other. In addition, the control points are always equal lengths. Symmetrical nodes produce the same curvature on both sides of the node.

Unless a curve changes direction sharply as it passes through a node, changing the node type will not noticeably affect the curve's shape. It will, however, affect the way you can reshape a curve.

You can close an open path by joining its two end nodes. You can also join end nodes on separate paths if the paths are all subpaths of the same object. You cannot join nodes of two separate objects. If you want to join nodes from separate curve objects, you must first combine them into a single curve object, then join the two end nodes.

{button ,AL(`CLS_Node')} **Related Topics**

Node.Segment

Property **Segment** As **Segment**

Description

The **Segment** property returns a value associated with the segment that contains the active node in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

This property returns a Read-Only value.

Example

The following code example displays the length, in document units, of the segment that contains the first node in the **Nodes** collection:

```
Sub Test()
MsgBox ActiveShape.Curve.Nodes(1).Segment.Length
End Sub
```

{button ,AL(`CLS_Node;FNC_Segment')} **Related Topics**

Node.GetDistanceFrom

Function **GetDistanceFrom**(ByVal **Node** As **Node**) As Double

Description

The **GetDistanceFrom** method returns the distance from the current node to a specified node in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Parameters	Description
Node	The **Node** parameter specifies a node, other than the active node, in a shape object. The distance from this node to the active node is returned with the **GetDistanceFrom** method.

{button ,AL(`CLS_Node;FNC_GetDistanceFrom')} **Related Topics**

Node.Application

Property **Application** AS **Application**

Node

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub AppVersion()
With ActiveShape.Curve.Nodes(1)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_Application')} **Related Topics**

Node.Parent

Property **Parent** AS **Shape**

Node

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

Example

The following code example displays the path of a node's parent shape object in a message box:

```
Sub ParentPath()
With ActiveShape.Curve.Nodes(1)
    MsgBox .Parent.Shpaes.Parent.Path
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_Parent')} **Related Topics**

Node.PositionX

Property **PositionX** AS Double

Node

Description

The **PositionX** property returns or sets a numerical value that indicates the horizontal placement of a node on a line or curve segment in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **PositionX** property returns or sets the X coordinate of a node in CorelDRAW. The **PositionX** property is set on the document page in relation to the page's reference point.

Example

The following code example displays the **X** and **Y** coordinates of the first node in a curve's subpath, using the **PositionX** and **PositionY** properties:

```
Sub Position()
With ActiveShape.Curve.Nodes(1)
    MsgBox .PositionX
    MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_PositionX')} **Related Topics**

Node.PositionY

Property **PositionY** AS Double

Node

Description

The **PositionY** property returns or sets a numerical value that indicates the vertical placement of a node on a line or curve segment in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **PositionY** property returns or sets the Y coordinate of a node in CorelDRAW. The **PositionY** property is set on the document page in relation to the page's reference point.

Example

The following code example displays the **X** and **Y** coordinates of the first node in a curve's subpath, using the **PositionX** and **PositionY** properties:

```
Sub Position()
With ActiveShape.Curve.Nodes(1)
    MsgBox .PositionX
    MsgBox .PositionY
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_PositionY')} **Related Topics**

Node.Type

Property **Type** AS **cdrNodeType**

Node

Description

The **Type** property returns or sets the node type in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

A node can be a cusp, smooth or symmetrical node. The **Type** property returns a value of cdrNodeType.

Example

The following code example converts all the cusp nodes in the curve object to smooth nodes.

```
Sub Test()
    Dim n As Node
    For Each n In ActiveShape.Curve.Nodes
        If n.Type = cdrCuspNode Then n.Type = cdrSmoothNode
    Next n
End Sub
```

{button ,AL(`CLS_Node;FNC_Type')} **Related Topics**

Node.SubPath

Property **SubPath** AS **SubPath**

Node

Description

The **SubPath** property returns the subpath containing a specific node in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

The **Subpath** property returns a Read-Only value.

Example

The following code example displays the length of a node's subpath in a message box. The length of the subpath is measured in document units:

```
Sub PathLength()
With ActiveShape.Curve.Nodes(1)
    MsgBox .Subpath.Length
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_SubPath')} **Related Topics**

Node.Index

Property **Index** AS Long

Node

Description

The **Index** property returns a value associated with a node object within its subpath in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

Nodes(5) refers to the fifth node in a subpath in CorelDRAW.

It is important to note that the **Index** property returns the index value of a node within a subpath, whereas the **AbsoluteIndex** property returns the index value of a node within a curve object (containing all subpaths). **Index** and **AbsoluteIndex** will return the same value if the node belongs to the first subpath of the curve.

The **Index** property returns a Read-Only value.

Example

The following code example loops through all the nodes of each subpath and prints the nodes' **Index** and **AbsoluteIndex** property in the Intermediate window of the Visual Basic Editor:

```
Sub Test()
    Dim s As Shape
    Dim sp As SubPath
    Dim n As Node
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        For Each sp In s.Curve.Subpaths
            For Each n In sp.Nodes
                Debug.Print n.Index, n.AbsoluteIndex
            Next n
        Next sp
    End If
End Sub
```

{button ,AL(`CLS_Node;FNC_Index')} **Related Topics**

Node.SubPathIndex

Property **SubPathIndex** AS Long

Node

Description

The **SubPathIndex** property returns the index value of the subpath within a curve object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **SubPathIndex** property returns a Read-Only value.

Example

The following code example displays the index value of a node's subpath in a message box:

```
Sub NodeSubpath()
With ActiveShape.Curve.Nodes(1)
    MsgBox .SubPathIndex
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_SubPathIndex')} **Related Topics**

Node.AbsoluteIndex

Property **AbsoluteIndex** AS Long

Node

Description

The **AbsoluteIndex** property returns the index value of a node within a curve object in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

It is important to note that the **Index** property returns the index value of a node within a subpath, whereas the **AbsoluteIndex** property returns the index value of a node within a curve object (containing all subpaths). **Index** and **AbsoluteIndex** will return the same value if the node belongs to the first subpath of the curve.

The **AbsoluteIndex** property returns a Read-Only value.

Example

The following code example loops through all the nodes of each subpath and prints the nodes' **Index** and **AbsoluteIndex** property in the Intermediate window of the Visual Basic Editor:

```
Sub Test()
    Dim s As Shape
    Dim sp As SubPath
    Dim n As Node
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        For Each sp In s.Curve.Subpaths
            For Each n In sp.Nodes
                Debug.Print n.Index, n.AbsoluteIndex
            Next n
        Next sp
    End If
End Sub
```

{button ,AL(`CLS_Node;FNC_AbsoluteIndex')} **Related Topics**

Node.IsEnding

Property **IsEnding** AS Boolean

Node

Description

The **IsEnding** property returns a True or False value indicating whether the node is the first or last node within its subpath in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

If the **IsEnding** property is True, the node is the last node of its subpath in CorelDRAW.

The **IsEnding** property returns a Read-Only value.

Example

The following code example marks each ending node in a curve object with a small circle, using the **IsEnding** property:

```
Sub Test()
    Dim n As Node
    For Each n In ActiveShape.Curve.Nodes
        If n.IsEnding Then
            ActiveLayer.CreateEllipse2 n.PositionX, n.PositionY, 0.1
        End If
    Next n
End Sub
```

{button ,AL(`CLS_Node;FNC_IsEnding')} **Related Topics**

Node.JoinWith

Sub **JoinWith**(ByRef **Target** AS **Node**)

Node

Description

The **JoinWith** method joins two nodes together in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

It is important to note that the **JoinWith** property merges two individual nodes into one, whereas the **ConnectWith** method connects two ending nodes by drawing a straight line between them.

In order to use the **JoinWith** method, you must pass the **Target** value as a parameter:

Parameters	Description
Target	The **Target** parameter identifies the node object that is joined to the specified node. For example, `Nodes.Item(2)` specifies the second node as the target node.

Example

The following code example joins the first and last nodes of the active curve object using the **JoinWith** method, where the curve has 9 nodes in total. This method merges the two points into one point:

```
Sub Connect()
Dim s as Shape
Dim sp as SubPath
Set s = ActiveShape
Set sp = ActiveShape.Curve.Subpaths(1)
With s.Curve.Nodes(1)
    .JoinWith sp.Nodes(9)
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_JoinWith')} **Related Topics**

Node.ConnectWith

Sub **ConnectWith**(ByRef **Target** AS **Node**)

Node

Description

The **ConnectWith** method connects a node to another node in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

It is important to note that the **JoinWith** property merges two individual nodes into one, whereas the **ConnectWith** method connects two ending nodes by drawing a straight line between them.

Both nodes must be ending nodes in order to use the **ConnectWith** method.

In order to use the **ConnectWith** method, you must pass the **Target** value as a parameter:

Parameters	Description
Target	The **Target** parameter identifies the node object that is connected to the specified node. For example, `Nodes.Item(2)` specifies the second node as the target node.

Example

The following code example connects the first and last nodes of the active curve object using the **ConnectWith** method, where the curve has 9 nodes in total. This method draws a line between the two points:

```
Sub Connect()
Dim s as Shape
Dim sp as SubPath
Set s = ActiveShape
Set sp = ActiveShape.Curve.Subpaths(1)
With s.Curve.Nodes(1)
    .ConnectWith sp.Nodes(9)
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_ConnectWith')} **Related Topics**

Node.BreakApart

Sub **BreakApart**()

Node

Description

The **BreakApart** method breaks a node into two unconnected nodes in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example breaks the curve into individual segments using the **BreakApart** method:

```
Sub Test()
    Dim s As Shape
    Dim i As Long
    Set s = ActiveShape
    If s.Type <> cdrCurveShape Then s.ConvertToCurves
    s.Curve.Closed = False
    For i = s.Curve.Nodes.Count To 1 Step -1
        s.Curve.Nodes(i).BreakApart
    Next i
    s.BreakApart
End Sub
```

{button ,AL(`CLS_Node;FNC_BreakApart')} **Related Topics**

Node.Delete

Sub **Delete**()

Node

Description

The **Delete** method removes a node from a curve object in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example deletes the first node in a curve's subpath, using the **Delete** method:

```
Sub Del()
With ActiveShape.Curve.Nodes(1)
    .Delete
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_Delete')} **Related Topics**

Node.GetPosition

Sub **GetPosition**(ByRef **PositionX** AS Double, ByRef **PositionY** AS Double)

Node

Description

The **GetPosition** method returns the horizontal and vertical position of a node object in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

It is important to note that the **GetPosition** method reads the coordinates of a node, whereas the **SetPosition** method moves the node to new coordinates.

In order to return the position of the node with the **GetPosition** method, you must pass the **X** and **Y** positions of the node as parameters:

Parameters	Description
PositionX	The **PositionX** parameter identifies the X, or horizontal, coordinate of a node in CorelDRAW. The X coordinate is measured in document units.
PositionY	The **PositionY** parameter identifies the Y, or vertical, coordinate of a node in CorelDRAW. The Y coordinate is measured in document units.

Example

The following code example reads the coordinates of the first node in a curve object, using the **GetPosition** method, and repositions the node in the curve by setting new coordinates with the **SetPosition** method:

```
Sub Position()
With ActiveShape.Curve.Nodes(1)
    .GetPosition 2, 5
    .SetPosition 2, 5
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_GetPosition')} **Related Topics**

Node.SetPosition

Sub **SetPosition**(ByVal **PositionX** AS Double, ByVal **PositionY** AS Double)

Node

Description

The **SetPosition** method sets the horizontal and vertical position of a node in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

It is important to note that the **GetPosition** method reads the coordinates of a node, whereas the **SetPosition** method moves the node to new coordinates.

In order to set the position of the node with the **SetPosition** method, you must pass the **X** and **Y** positions of the node as parameters:

Member of **Node**

Parameters	Description
PositionX	The **PositionX** parameter identifies the X, or horizontal, coordinate of a node in CorelDRAW. The X coordinate is measured in document units.
PositionY	The **PositionY** parameter identifies the Y, or vertical, coordinate of a node in CorelDRAW. The Y coordinate is measured in document units.

Example

The following code example reads the coordinates of the first node in a curve object, using the **GetPosition** method, and repositions the node in the curve by setting new coordinates with the **SetPosition** method:

```
Sub Position()
With ActiveShape.Curve.Nodes(1)
    .GetPosition 2, 5
    .SetPosition 2, 5
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_SetPosition')} **Related Topics**

Node.Move

Sub **Move**(ByVal DeltaX AS Double, ByVal DeltaY AS Double)

Node

Description

The **Move** method changes the position of a node in a curve object by moving it a specified distance from its original location in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

In order to move a node with the **Move** method, you must pass the **DeltaX** and **DeltaY** values as parameters:

Parameters	Description
DeltaX	The **DeltaX** parameter specifies the horizontal distance to move a node with the **Move** method. Delta refers to a limited incremental value in a variable. This value is optional and the default value is 0.
DeltaY	The **DeltaY** parameter specifies the vertical distance to move a node with the **Move** method. Delta refers to a limited incremental value in a variable. This value is optional and the default value is 0.

Example

The following code example moves the first node in the active curve shape, using the **Move** method:

```
Sub NodeMove()
With ActiveShape.Curve.Nodes(1)
    .Move 1, 1
End With
End Sub
```

{button ,AL(`CLS_Node;FNC_Move')} **Related Topics**

Node.Next

Function **Next**() AS **Node**

Node

Description

The **Next** method returns the node in the **Nodes** collection that follows the selected node object in CorelDRAW. AA node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

For example, if the selected node has an index value of 2, the **Next** method returns the node with an index value of 3 in the **Nodes** collection.

Example

The following code example changes the node type of the third node to smooth in the active shape object, if the fourth node type is symmetrical, and the third node type is cusp:

```
Sub Test()
    Dim s As Shape
    Dim n As Node
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        Set n = s.Curve.Nodes(4)
        If n.Type = cdrSymmetricalNode And n.Previous.Type = cdrCuspNode Then
            n.Previous.Type = cdrSmoothNode
        End If
    End If
End Sub
```

{button ,AL(`CLS_Node;FNC_Next')} **Related Topics**

Node.Previous

Function **Previous**() AS **Node**

Node

Description

The **Previous** method returns the node in the **Nodes** collection that precedes the selected node object in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

For example, if the selected node object has an index value of 2, the **Previous** method returns the node with an index value of 1 in the **Nodes** collection.

Example

The following code example changes the node type of the fifth node to smooth in the active shape object, if the fourth node type is symmetrical, and the fifth node type is cusp:

```
Sub Test()
    Dim s As Shape
    Dim n As Node
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        Set n = s.Curve.Nodes(4)
        If n.Type = cdrSymmetricalNode And n.Next.Type = cdrCuspNode Then
            n.Next.Type = cdrSmoothNode
        End If
    End If
End Sub
```

{button ,AL(`CLS_Node;FNC_Previous')} **Related Topics**

Nodes properties

Nodes Legend

- Application
- Count
-
- Item
- Parent

Nodes methods

Nodes Legend

- All
- AllExcluding
- Range

Nodes

Class **Nodes**

Properties Methods Referenced by

The **Nodes** class defines the characteristics of node collection objects and describes the look and behavior of the objects through its properties and methods.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the type of node (smooth, cusp, or symmetrical) and segment (line or curve) selected.

There are three types of nodes and the control points of each node type behave differently:

Cusp node

The control points of a cusp node move independently from one another. A curve that passes through a cusp node can bend at a sharp angle.

Smooth node

The control points of a smooth node are always directly opposite each other. When you move one control point, the other moves also. The control points of a smooth node may be of different lengths. Smooth nodes produce a smooth transition between line segments.

Symmetrical node

The control points of a symmetrical node are always directly opposite each other. In addition, the control points are always equal lengths. Symmetrical nodes produce the same curvature on both sides of the node.

Unless a curve changes direction sharply as it passes through a node, changing the node type will not noticeably affect the curve's shape. It will, however, affect the way you can reshape a curve.

You can close an open path by joining its two end nodes. You can also join end nodes on separate paths if the paths are all subpaths of the same object. You cannot join nodes of two separate objects. If you want to join nodes from separate curve objects, you must first combine them into a single curve object, then join the two end nodes.

{button ,AL(`CLS_Nodes')} **Related Topics**

Control points are points that extend from nodes along curves that are being edited with the Shape tool. Control points determine the angle at which the curve passes through the node. Control points appear when you select a node or segment using the Shape tool.

A curve is an object that can be any shape. Curve objects have nodes (the points on a path that determine its shape) and control points (points that extend from nodes to further define a path's shape) that you manipulate to change the object's shape. Curve objects can be drawn with the Freehand tool, Bezier tool, Spiral tool, and Natural Pen tool. You can also convert text and objects drawn with the Rectangle tool, Ellipse tool, and Polygon tool into curve objects by using the Convert To Curves command in the Arrange menu.

cdrCuspNode=0

cdrSmoothNode=1

cdrSymmetricalNode=2

cdrMixedNodes=3

Nodes.Application

Property **Application** AS **Application**

Nodes

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub NodesApp()
With ActiveShape.Curve.Nodes
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_Application')} **Related Topics**

Nodes.Parent

Property **Parent** AS **Shape**

Nodes

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the **Node** collection's parent curve object in a message box. The length is measured in document units:

```
Sub NodesParent()
With ActiveShape.Curve.Nodes
    MsgBox .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_Parent')} **Related Topics**

Nodes.Item

Property **Item**(ByVal **Index** AS Long) AS **Node**

Nodes

Description

The **Item** property returns a value associated with the index number of a node object in the **Nodes** collection of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

`Nodes(2)` refers to the second node object in the **Nodes** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Nodes.Item(2)` is the same as `Nodes(2)` - they both reference the second node object in the **Nodes** collection.

You must reference a node in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Nodes** collection; it uniquely identifies each member of the collection.

Example

The following code example sets the node type of the first node in the **Nodes** collection of CorelDRAW. The node type is set to a value equal to a smooth node. The control points of a smooth node are always directly opposite each other. When you move one control point, the other moves also. The control points of a smooth node may be of different lengths. Smooth nodes produce a smooth transition between line segments.

```
Sub NodesItem()
With ActiveShape.Curve.Nodes
    .Item(1).Type = cdrSmoothNode
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_Item')} **Related Topics**

Nodes.Count

Property **Count** AS Long

Nodes

Description

The **Count** property returns the number of nodes in the **Nodes** collection of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of nodes in the **Nodes** collection in a message box:

```
Sub NodesCount()
With ActiveShape.Curve.Nodes
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_Count')} **Related Topics**

Nodes.Range

Function **Range**(ByVal **IndexArray** AS Variant) AS **NodeRange**

Nodes

Description

The **Range** method returns or sets a value associated with a series of node objects in the **Nodes** collection in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

The **Range** method allows you to reference several nodes concurrently. The use of an index range makes it easier and faster to find, sort, and edit multiple nodes in the **Nodes** collection.

You must reference the nodes in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify specific nodes in the **Nodes** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific node objects that become members of the array.

Example

The following code example deletes an array of nodes in the **Nodes** collection in CorelDRAW. The array consists of the first and the third nodes of the collection. A message box displays the number of nodes before and after the deletion to show you that the nodes are deleted from the collection. Deleting nodes will alter the appearance of the active shape:

```
Sub NodesRange()

With ActiveShape.Curve.Nodes
    MsgBox "There are " & .Count & " nodes in the collection."
    .Range(1, 2).Delete
    MsgBox "There are " & .Count & " nodes in the collection."
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_Range')} **Related Topics**

Nodes.All

Function **All**() AS **NodeRange**

Nodes

Description

The **All** method returns or sets a value associated with all node objects in the **Nodes** collection in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

The **All** method allows you to reference all nodes concurrently. The use of the **All** method makes it easier and faster to find, sort, and edit all nodes in the **Nodes** collection.

Example

The following code example sets the node type of all the nodes in the **Nodes** collection to symmetrical nodes. The control points of a symmetrical node are always directly opposite each other. The control points are always equal lengths. Symmetrical nodes produce the same curvature on both sides of the node.

```
Sub NodesAll()
With ActiveShape.Curve.Nodes
    .All.SetType cdrSymmetricalNode
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_All')} **Related Topics**

Nodes.AllExcluding

Function **AllExcluding**(ByVal **IndexArray** AS Variant) AS **NodeRange**

Nodes

Description

The **AllExcluding** method returns or sets a value associated with a series of node objects in the **Nodes** collection in CorelDRAW, with the exception of the nodes identified by their index numbers in the method parameters. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

You must reference the nodes in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify and group specific nodes in the **Nodes** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific node objects that become members of the array.

Example

The following code example deletes all nodes in the **Nodes** collection, with the exception of the first and third nodes. Deleting nodes will alter the appearance of the active shape:

```
Sub NodesAllEx()
With ActiveShape.Curve.Nodes
    .AllExcluding (1,3).Delete
End With
End Sub
```

{button ,AL(`CLS_Nodes;FNC_AllExcluding')} **Related Topics**

NodeRange properties

NodeRange Legend

- Application
- Count
-
- Item
- Parent
- PositionX
- PositionY
- SegmentRange
- SizeHeight
- SizeWidth
- Type

NodeRange methods

NodeRange Legend

- Add
- AddRange
- AutoReduce
- BreakApart
- Delete
- Move
- Remove
- RemoveAll
- Rotate
- SetType
- Skew
- Smoothen
- Stretch

NodeRange

Class **NodeRange**

Properties Methods Referenced by

The **NodeRange** class defines the characteristics of a range of node objects and describes the look and behavior of the object range through its properties and methods.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the type of node (smooth, cusp, or symmetrical) and segment (line or curve) selected.

There are three types of nodes and the control points of each node type behave differently:

Cusp node

The control points of a cusp node move independently from one another. A curve that passes through a cusp node can bend at a sharp angle.

Smooth node

The control points of a smooth node are always directly opposite each other. When you move one control point, the other moves also. The control points of a smooth node may be of different lengths. Smooth nodes produce a smooth transition between line segments.

Symmetrical node

The control points of a symmetrical node are always directly opposite each other. In addition, the control points are always equal lengths. Symmetrical nodes produce the same curvature on both sides of the node.

Unless a curve changes direction sharply as it passes through a node, changing the node type will not noticeably affect the curve's shape. It will, however, affect the way you can reshape a curve.

You can close an open path by joining its two end nodes. You can also join end nodes on separate paths if the paths are all subpaths of the same object. You cannot join nodes of two separate objects. If you want to join nodes from separate curve objects, you must first combine them into a single curve object, then join the two end nodes.

{button ,AL(`CLS_NodeRange')} **Related Topics**

NodeRange.SegmentRange

Property **SegmentRange** As **SegmentRange**

Description

The **SegmentRange** property returns a value associated with a range of segments that correspond to a node range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **SegmentRange** property returns a Read-Only value.

{button ,AL(`CLS_NodeRange;FNC_SegmentRange')} **Related Topics**

NodeRange.AddRange

Sub **AddRange**(ByVal **NodeRange** As **NodeRange**)

Description

The **AddRange** method adds a node range to an existing node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

In order the use the **AddRange** method, you must pass the **NodeRange** value as a parameter:

Parameters	Description
NodeRange	The **NodeRange** parameter specifies the range of node items that are added to the current node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example creates two separate node ranges from the active shape, using the **Range** and **AllExluding** methods. Both node ranges are combined using the **AddRange** method. The number of nodes is displayed in a message box:

```
Sub Range()
Dim n as NodeRange
Dim NewRange as New NodeRange
Set NewRange = ActiveShape.Curve.Nodes.Range (1, 2)
'creates a range with the first and second node in the shape
Set n = ActiveShape.Curve.Nodes.AllExcluding (1, 2)
'creates a range with all nodes except the first and second nodes
With n
    .AddRange NewRange
    .MsgBox .NodeRange.Count
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_AddRange')} **Related Topics**

NodeRange.BreakApart

Sub **BreakApart**()

Description

The **BreakApart** method breaks a curve at each node in a node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

{button ,AL(`CLS_NodeRange;FNC_BreakApart')} **Related Topics**

NodeRange.RemoveAll

Sub **RemoveAll**()

Description

The **RemoveAll** method removes all nodes from a node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example deletes all nodes in a node range in the active shape of CorelDRAW:

```
Sub SegType()
Dim n as NodeRange
Set n = ActiveShape.Curve.Nodes.All
'creates a range with all nodes in the shape
With n
    .RemoveAll
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_RemoveAll')} **Related Topics**

NodeRange.SetType

Sub **SetType**(ByVal **Type** As **cdrNodeType**)

Description

The **SetType** method sets the type of all nodes within a node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **SetType** method returns a value of cdrNodeType.

In order to use the **SetType** method, you must pass the **Type** value as a parameter:

Parameters	Description
Type	The **Type** parameter specifies the node type for all the nodes within a node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example makes all the nodes in the active shape smooth nodes, using the **SetType** method:

```
Sub NodeType()
Dim n as NodeRange
Set n = ActiveShape.Curve.Nodes.All
'creates a range with all nodes in the shape
With n
    .SetType cdrSmoothNode
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_SetType')} **Related Topics**

NodeRange.Smoothen

Sub **Smoothen**(ByVal **Smoothness** As Long)

Description

The **Smoothen** method smoothens the nodes in a node range in a shape object of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Parameters	Description
Smoothness	The **Smoothness** parameter sets the level of smoothness in a node range in CorelDRAW. Higher values indicate an increased level of smoothness.

{button ,AL(`CLS_NodeRange;FNC_Smoothen')} **Related Topics**

NodeRange.Application

Property **Application** AS **Application**

NodeRange

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub AppVersion()
With ActiveShape.Curve.Nodes.Range
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Application')} **Related Topics**

NodeRange.Parent

Property **Parent** AS **Shape**

NodeRange

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

Example

The following code example displays the length of the node range's parent curve object, measured in document units:

```
Sub LengthParent()
With ActiveShape.Curve.Nodes.Range
    .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Parent')} **Related Topics**

NodeRange.Item

Property **Item**(ByVal **Index** AS Long) AS **Node**

NodeRange

Description

The **Item** property returns a value associated with the index number of a **NodeRange** object in the **NodeRange** class of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

`NodeRange(2)` refers to the second node object in the **NodeRange** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the range. For example, `NodeRange.Item(2)` is the same as `NodeRange(2)` - they both reference the second node range object in the range.

You must reference a member of a node range by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **NodeRange** class: it uniquely identifies each member of the node range.

Example

The following code example creates a node range from all nodes in the curve, and removes all symmetrical and smooth nodes from the range (but the nodes are not deleted from the curve). The remaining collection of cusp nodes are rotated by 30°, relative to its geometric center of the active shape:

```
Sub Test()
    Dim nr As NodeRange
    Dim n As Node
    Dim i As Long
    Set nr = ActiveShape.Curve.Nodes.All
    For i = nr.Count To 1 Step -1
        If nr.Item(i).Type <> cdrCuspNode Then nr.Remove i
        ' The same as: If nr(i).Type <> cdrCuspNode Then nr.Remove i
    Next i
    ActiveDocument.ReferencePoint = cdrCenter
    If nr.Count > 0 Then nr.Rotate 30
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Item')} **Related Topics**

NodeRange.Count

Property **Count** AS Long

NodeRange

Description

The **Count** property returns the number of nodes in the **NodeRange** class of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of selected nodes in the active shape of CorelDRAW:

```
Sub Test()
    Dim nr As NodeRange
    Dim s As String
    Set nr = ActiveShape.Curve.Selection
    If nr.Count = 0 Then
        s = "No nodes selected"
    MsgBox s
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Count')} **Related Topics**

NodeRange.Type

Property **Type** AS **cdrNodeType**

NodeRange

Description

The **Type** property returns a value associated with a node type in the **NodeRange** class of CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

If you are using the **Type** property for the entire node range, a value is returned only if the node type is common to all nodes included within the range. The **Type** property can also be used with the **Item** property to return the node type for a specific node within the range. If there are nodes of different types, the **Type** property will return cdrMixedNodes.

The **Type** property returns a value of cdrNodeType.

Example

The following code example displays the node type of the node range in the active shape of CorelDRAW:

```
Sub RangeNodes()
With ActiveShape.Curve.Nodes.Range
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Type')} **Related Topics**

NodeRange.PositionX

Property **PositionX** AS Double

NodeRange

Description

The **PositionX** property returns the horizontal position of the node that is located farthest left within the node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **PositionX** property is equivalent to the position of the bounding box that encloses all the nodes in the node range. The position is relative to the document reference point.

It is important to note that **PositionX** and **PositionY** are always 0 if there are no nodes in the node range.

The **PositionX** property returns a Read-Only value and is measured in document units.

Example

The following code example draws a rectangle around the selected nodes in the node range of the active shape:

```
Sub Test()
    Dim nr As NodeRange
    Dim x1 As Double, y1 As Double
    Dim x2 As Double, y2 As Double
    Set nr = ActiveShape.Curve.Selection
    ActiveDocument.ReferencePoint = cdrTopLeft
    x1 = nr.PositionX
    y1 = nr.PositionY
    ActiveDocument.ReferencePoint = cdrBottomRight
    x2 = nr.PositionX
    y2 = nr.PositionY
    ActiveLayer.CreateRectangle x1, y1, x2, y2
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_PositionX')} **Related Topics**

NodeRange.PositionY

Property **PositionY** AS Double

NodeRange

Description

The **PositionY** property returns the vertical position of the node that is located farthest left within the node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **PositionY** property is equivalent to the position of the bounding box that encloses all the nodes in the node range. The position is relative to the document reference point.

It is important to note that **PositionX** and **PositionY** are always 0 if there are no nodes in the node range.

The **PositionY** property returns a Read-Only value and is measured in document units.

Example

The following code example draws a rectangle around the selected nodes in the node range of the active shape:

```
Sub Test()
    Dim nr As NodeRange
    Dim x1 As Double, y1 As Double
    Dim x2 As Double, y2 As Double
    Set nr = ActiveShape.Curve.Selection
    ActiveDocument.ReferencePoint = cdrTopLeft
    x1 = nr.PositionX
    y1 = nr.PositionY
    ActiveDocument.ReferencePoint = cdrBottomRight
    x2 = nr.PositionX
    y2 = nr.PositionY
    ActiveLayer.CreateRectangle x1, y1, x2, y2
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_PositionY')} **Related Topics**

NodeRange.SizeWidth

Property **SizeWidth** AS Double

NodeRange

Description

The **SizeWidth** property returns a value that indicates the horizontal distance, or width, between the leftmost and rightmost nodes in a node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

This property specifies the width of the bounding box containing all of the nodes in the node range.

The **SizeWidth** property returns a Read-Only value and is measured in document units.

Example

The following code example draws a rectangle around the selected nodes in the node range of the active shape:

```
Sub Test()
    Dim nr As NodeRange
    Dim x1 As Double, y1 As Double
    Dim x2 As Double, y2 As Double
    Set nr = ActiveShape.Curve.Selection
    ActiveDocument.ReferencePoint = cdrBottomLeft
    x1 = nr.PositionX
    y1 = nr.PositionY
    x2 = x1 + nr.SizeWidth
    y2 = y1 + nr.SizeHeight
    ActiveLayer.CreateRectangle x1, y1, x2, y2
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_SizeWidth')} **Related Topics**

NodeRange.SizeHeight

Property **SizeHeight** AS Double

NodeRange

Description

The **SizeHeight** property returns a value that indicates the vertical distance, or height, between the topmost and bottommost nodes in a node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

This property specifies the height of the bounding box containing all of the nodes in the node range.

The **SizeWidth** property returns a Read-Only value and is measured in document units.

Example

The following code example draws a rectangle around the selected nodes in the node range of the active shape:

```
Sub Test()
    Dim nr As NodeRange
    Dim x1 As Double, y1 As Double
    Dim x2 As Double, y2 As Double
    Set nr = ActiveShape.Curve.Selection
    ActiveDocument.ReferencePoint = cdrBottomLeft
    x1 = nr.PositionX
    y1 = nr.PositionY
    x2 = x1 + nr.SizeWidth
    y2 = y1 + nr.SizeHeight
    ActiveLayer.CreateRectangle x1, y1, x2, y2
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_SizeHeight')} **Related Topics**

NodeRange.Add

Sub **Add**(ByRef **Node** AS **Node**)

NodeRange

Description

The **Add** method places a specified node object into an existing node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Adding a node to a node range doesn't make that node active within a curve object - it merely includes the node in future references to the node range.

You must reference a member of a node range by passing the node object to be added as a parameter:

Parameters	Description
Node	The **Node** parameter identifies the node object that is added to the node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example creates a node range containing the 1st, 2nd and 3rd nodes of the curve, then adds a 5th node to the curve and deletes all previous four nodes from the curve object:

```
Sub Test()
    Dim nr As NodeRange
    Set nr = ActiveShape.Curve.Nodes.Range(Array(1, 2, 3))
    nr.Add ActiveShape.Curve.Nodes(5)
    nr.Delete
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Add')} **Related Topics**

NodeRange.Remove

Sub **Remove**(ByVal **Index** AS Long)

NodeRange

Description

The **Remove** method takes a specified node object out of an existing node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Removing a node takes it out of the node range, but it doesn't delete the node from a curve object.

You must reference a member of a node range by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in the **NodeRange** class; it uniquely identifies each member of the node range. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Example

The following code example gets the selected nodes, removes the first and last selected node from the range and deletes the remaining nodes. As a result, all the selected nodes, except the first and last, are deleted from the curve:

```
Sub Test()
    Dim nr As NodeRange
    Set nr = ActiveShape.Curve.Selection
    nr.Remove 1
    nr.Remove nr.Count
    nr.Delete
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Remove')} **Related Topics**

NodeRange.Move

Sub **Move**(ByVal DeltaX AS Double, ByVal DeltaY AS Double, ByVal AnchorIndex AS Long, ByVal ElasticMode AS Boolean)

NodeRange

Description

The **Move** method repositions all the nodes within a node range, moving the node a specified distance in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

In order to use the **Move** method, you must pass several values as parameters:

Parameters	Description
DeltaX	The **DeltaX** parameter specifies the horizontal distance to move a node with the **Move** method. Delta refers to a limited incremental value in a variable. This value is optional and the default value is 0.
DeltaY	The **DeltaY** parameter specifies the vertical distance to move a node with the **Move** method. Delta refers to a limited incremental value in a variable. This value is optional and the default value is 0.
AnchorIndex	The **AnchorIndex** parameter identifies the node in the node range that acts as the anchor when using the **Move** method. It returns the index number of the node. An anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object.
ElasticMode	The **ElasticMode** parameter returns a True or False value that indicates if elastic mode is enabled while moving a node in the node range. If **ElasticMode** is True, only the node identified by the **AnchorIndex** parameter is moved by the exact offset specified. This value is optional and the default value is False.

Example

The following code example creates a node range from all nodes in the curve except for the first and the last nodes. The nodes are then moved vertically by 1" (where inches is the document unit) and moved back down by 1" with elastic mode enabled. The middle node (node_count/2) is used in the last operation as the anchor node:

```
Sub Test()
    Dim crv As Curve
    Dim nr As NodeRange
    Set crv = ActiveShape.Curve
    Set nr = crv.Nodes.AllExcluding(Array(1, crv.Nodes.Count))
    nr.Move 0, 1
    nr.Move 0, -1, nr.Count \ 2, True
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Move')} **Related Topics**

NodeRange.Delete

Sub **Delete**()

NodeRange

Description

The **Delete** method removes all the nodes within a node range in CorelDRAW. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When using the **Delete** method, the nodes are physically removed from the curve object.

Example

The following code example deletes all of the selected nodes in the active shape curve object:

```
Sub Test()
    ActiveShape.Curve.Selection.Delete
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Delete')} **Related Topics**

NodeRange.Stretch

Sub **Stretch**(ByVal RatioX AS Single, ByVal RatioY AS Single, ByVal UseReferencePoint AS Boolean, ByVal StretchAnchorX AS Double, ByVal StretchAnchorY AS Double)

NodeRange

Description

The **Stretch** property resizes the nodes within a node range as if the node range is a separate curve object. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Stretching an object means to size an object horizontally or vertically. Stretching changes the size of an object in one direction only, as opposed to sizing, which maintains the aspect ratio (the ratio of height to width).

In order to use the **Stretch** method, you must pass several values as parameters:

Parameters	Description
RatioX	The **RatioX** parameter specifies the width value of the aspect ratio used when stretching objects. The aspect ratio is the ratio of the width of an image to its height (expressed mathematically as x:y). For example, the aspect ratio of an image that is 640 x 480 pixels is 4:3. This value is optional and the default value is 100.
RatioY	The **RatioY** parameter specifies the height value of the aspect ratio used when stretching objects. The aspect ratio is the ratio of the width of an image to its height (expressed mathematically as x:y). For example, the aspect ratio of an image that is 640 x 480 pixels is 4:3. This value is optional and the default value is 100.
UseReferencePoint	The **UseReferencePoint** parameter is a True or False value that specifies if the document's reference point is used when stretching the node range. This value is optional and the default value is True.
StretchAnchorX	The **StretchAnchorX** parameter specifies the horizontal position of the anchor point when stretching the node range. The anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object. This value is optional and the default value is 0. This value is measured in document units.
StretchAnchorY	The **StretchAnchorY** parameter specifies the vertical position of the anchor point when stretching the node range. The anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object. This value is optional and the default value is 0. This value is measured in document units.

Example

The following code example stretches the node range in the active shape with ratio values of 100, no reference point and the anchor point set at (0, 0):

```
Sub SkewNode()
Dim nr as NodeRange
Set nr = ActiveShape.Curve.Nodes.All
With nr
    .Stretch 100, 100, False, 0, 0
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Stretch')} **Related Topics**

NodeRange.Rotate

Sub **Rotate**(ByVal **Angle** AS Double, ByVal UseReferencePoint AS Boolean, ByVal RotationCenterX AS Double, ByVal RotationCenterY AS Double)

NodeRange

Description

The **Rotate** property rotates the nodes within a node range as if the node range is a separate curve object. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Rotating an object means to reposition and reorient the object by turning it around its center of rotation.

In order to use the **Rotate** method, you must pass several values as parameters:

Parameters	Description
Angle	The **Angle** parameter specifies the angle, measured in degrees, that the node range is rotated. Angle = 45 rotates the node range 45.
UseReferencePoint	The **UseReferencePoint** parameter is a True or False value that specifies if the document's reference poin is used when stretching the node range. This value is optional and the default value is True.
RotateAnchorX	The **RotateAnchorX** parameter specifies the horizontal position of the rotation point when stretching the node range. The rotation point is center of rotation around which the node range rotates when using the **Rotate** method. This value is optional and the default value is 0. This value is measured in document units.
RotateAnchorY	The **RotateAnchorY** parameter specifies the vertical position of the rotation point when stretching the node range. The rotation point is center of rotation around which the node range rotates when using the **Rotate** method. This value is optional and the default value is 0. This value is measured in document units.

Example

The following code example rotates the selected nodes around their geometrical center in the active shape. The nodes are rotated 30 degrees:

```
Sub Test()
    Dim nr As NodeRange
    Set nr = ActiveShape.Curve.Selection
    ActiveDocument.ReferencePoint = cdrCenter
    nr.Rotate 30
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Rotate')} **Related Topics**

NodeRange.Skew

Sub **Skew**(ByVal AngleX AS Double, ByVal AngleY AS Double, ByVal UseReferencePoint AS Boolean, ByVal SkewAnchorX AS Double, ByVal SkewAnchorY AS Double)

NodeRange

Description

The **Skew** property rotates the nodes within a node range as if the node range is a separate curve object. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

In order to use the **Skew** method, you must pass several values as parameters:

Parameters	Description
AngleX	The **AngleX** parameter specifies the degree in which to slant the horizontal lines of a node range's object. This value is optional and the default value is 0.
AngleY	The **AngleY** parameter specifies the degree in which to slant the vertical lines of a node range's object. This value is optional and the default value is 0.
UseReferencePoint	The **UseReferencePoint** parameter is a True or False value that specifies if the document's reference poin is used when stretching the node range. This value is optional and the default value is True.
SkewAnchorX	The **SkewAnchorX** parameter specifies the horizontal position of the anchor point when skewing the node range. The anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object. This value is optional and the default value is 0. This value is measured in document units.
SkewAnchorY	The **SkewAnchorY** parameter specifies the vertical position of the anchor point when skewing the node range. The anchor point is the point that remains stationary when you stretch, scale, mirror, or skew an object. This value is optional and the default value is 0. This value is measured in document units.

Example

The following code example skews the node range in the active shape horizontally and vertically by 25. A reference point is not set and the anchor point is set at (0, 0):

```
Sub SkewNode()
Dim nr as NodeRange
Set nr = ActiveShape.Curve.Nodes.All
With nr
    .Skew 25, 25, False, 0, 0
End With
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_Skew')} **Related Topics**

NodeRange.AutoReduce

Sub **AutoReduce**(ByVal **PrecisionMargin** AS Double)

NodeRange

Description

The **AutoReduce** method increases the roundness of a node range's curve object by removing extra nodes, keeping the shape of the curve within a given deviation margin. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

In order to use the **AutoReduce** method, you must pass the **PrecisionMargin** value as a parameter:

Parameters	Description
PrecisionMargin	The **PrecisionMargin** parameter specifies the maximum allowable deviation of the curve shape, measured in document units of length. Specifying larger numbers will leave fewer nodes and may cause larger distortion to the shape.

Example

The following code example removes any extra nodes in the active shape, using the **AutoReduce** method, keeping the curve shape changes within 0.01", where inches is the document unit measurement:.

```
Sub Test()
    ActiveShape.Curve.Nodes.All.AutoReduce 0.01
End Sub
```

{button ,AL(`CLS_NodeRange;FNC_AutoReduce')} **Related Topics**

Segment properties

Segment Legend

- AbsoluteIndex

- Application
  - EndingControlPointAngle
  - EndingControlPointLength
  - EndingControlPointX
  - EndingControlPointY

- EndNode

- Index

- Length

- Parent
  - StartingControlPointAngle
  - StartingControlPointLength
  - StartingControlPointX
  - StartingControlPointY

- StartNode

- SubPath

- SubPathIndex
  - Type

Segment methods

Segment Legend

- AddNodeAt
- BreakApartAt
- FindParamOffset
- GetAbsoluteOffset
- GetBendPoints
- GetCurvatureAt
- GetCurveSpeedAt
- GetIntersections
- GetPeaks
- GetPerpendicularAt
- GetPointPositionAt
- GetTangentAt
- Next
- Previous

Segment

Class **Segment**

Properties Methods Referenced by

The **Segment** class defines the characteristics of segment objects and describes the look and behavior of the collection objects through its properties and methods.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

You can also change the direction of a curve and smooth out a curve using buttons on the Shape tool Property Bar.

{button ,AL(`CLS_Segment')} **Related Topics**

Segment.Application

Property **Application** AS **Application**

Segment

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_Application')} **Related Topics**

Segment.Parent

Property **Parent** AS **Shape**

Segment

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the first segment's parent curve in CorelDRAW::

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_Parent')} **Related Topics**

Segment.Type

Property **Type** AS **cdrSegmentType**

Segment

Description

The **Type** property returns or sets the segment type of a segment in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

The **Type** property returns cdrSegmentType.

Example

The following code example displays a numerical value that indicates the segment type of the first segment in the active shape:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_Type')} **Related Topics**

Segment.SubPath

Property **SubPath** AS **SubPath**

Segment

Description

The **SubPath** property returns the subpath containing a segment in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

The **Subpath** property returns a Read-Only value.

Example

The following code example counts the number of nodes on the subpath in the fist segment of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .SubPath.Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_SubPath')} **Related Topics**

Segment.Length

Property **Length** AS Double

Segment

Description

The **Length** property returns the length of a selected segment in CoreDRAW. Length is measured in document units, which may vary from document to document, and returns a value of cdrUnit.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

The **Length** property returns a Read-Only value.

Example

The following code example displays the length of the first segment in the active shape, in document units, in a message box:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .Length
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_Length')} **Related Topics**

Segment.Index

Property **Index** AS Long

Segment

Description

The **Index** property returns a value associated with a segment in the **Segments** collection of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

Segments(5) refers to the fifth segment in the collection.

The **Index** property returns a Read-Only value.

Example

The following code example uses the index value of 1 to identify the first segment in the active shape. The length of the first segment in the active shape, in document units, displays in a message box:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .Length
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_Index')} **Related Topics**

Segment.SubPathIndex

Property **SubPathIndex** AS Long

Segment

Description

The **SubPathIndex** property returns the index value of the subpath within a segment in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

The **SubPathIndex** property returns a Read-Only value.

Example

The following code example displays the index value of the segment's subpath object in a message box:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .SubPathIndex
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_SubPathIndex')} **Related Topics**

Segment.AbsoluteIndex

Property **AbsoluteIndex** AS Long

Segment

Description

The **AbsoluteIndex** property returns the index value of a segment within a curve object in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

It is important to note that the **Index** property returns the index value of a segment within a subpath, whereas the **AbsoluteIndex** property returns the index value of a node within a curve object (containing all subpaths). **Index** and **AbsoluteIndex** will return the same value if the node belongs to the first subpath of the curve.

The **AbsoluteIndex** property returns a Read-Only value.

Example

The following code example displays the absolute index value of the first segment in the active shape:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .AbsoluteIndex
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_AbsoluteIndex')} **Related Topics**

Segment.StartingControlPointLength

Property **StartingControlPointLength** AS Double

Segment

Description

The **StartingControlPointLength** property returns or sets the length of the starting control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

The length is measured in document units. The length and angle of the control point dictate the appearance of the segment.

Example

The following code example sets the starting and ending control point length for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointLength = 5
    .EndingControlPointLength = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_StartingControlPointLength')} **Related Topics**

Segment.StartingControlPointAngle

Property **StartingControlPointAngle** AS Double

Segment

Description

The **StartingControlPointAngle** property returns or sets the angle, in degrees, of the starting control point of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

You can control the curve of a segment by varying the control point's angle.

Example

The following code example sets the starting and ending control point angle for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointAngle = 45
    .EndingControlPointAngle = 45
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_StartingControlPointAngle')} **Related Topics**

Segment.EndingControlPointLength

Property **EndingControlPointLength** AS Double

Segment

Description

The **EndingControlPointLength** property returns or sets the length of the ending control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

The length is measured in document units. The length and angle of the control point dictate the appearance of the segment.

Example

The following code example sets the starting and ending control point length for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointLength = 5
    .EndingControlPointLength = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_EndingControlPointLength')} **Related Topics**

Segment.EndingControlPointAngle

Property **EndingControlPointAngle** AS Double

Segment

Description

The **StartingControlPointAngle** property returns or sets the angle, in degrees, of the starting control point of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

You can control the curve of a segment by varying the control point's angle.

Example

The following code example sets the starting and ending control point angle for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointAngle = 45
    .EndingControlPointAngle = 45
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_EndingControlPointAngle')} **Related Topics**

Segment.GetPointPositionAt

Sub **GetPointPositionAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**, ByRef **X** AS Double, ByRef **Y** AS Double)

Segment

Description

The **GetPointPositionAt** method returns the coordinates of a point located on a segment in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

In order to return a point position with the **GetPointPositionAt** method, you must pass the **Offset**, **OffsetType**, **X** and **Y** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).
X	The **X** parameter specifies the horizontal position of a point on a segment's subpath. The X coordinate is measured in document units,.
Y	The **Y** parameter specifies the vertical position of a point on a segment's subpath. The Y coordinateis measured in document units,.

Example

The following code example gets the point position at coordinate (2, 2) in the first segment of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetPointPositionAt (1, cdrRelativeSegmentOffset, 2, 2)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetPointPositionAt')} **Related Topics**

Segment.BreakApartAt

Function **BreakApartAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **Node**

Segment

Description

The **BreakApartAt** method separates a segment into two unconnected segments in CorelDRAW. A segment is the part of a curve lying between two nodes. Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

In order to break a segment with the **BreakApartAt** method, you must pass the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

{button ,AL(`CLS_Segment;FNC_BreakApartAt')} **Related Topics**

Segment.AddNodeAt

Function **AddNodeAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **Node**

Segment

Description

The **AddNodeAt** method inserts a node on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you add a new node to a segment with the **AddNodeAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example adds a new node to the first segment of the active shape in CorelDRAW, offset by 1:

```
SubApp()
With ActiveShape.Curve.Segments(1)
    .AddNodeAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_AddNodeAt')} **Related Topics**

Segment.GetPerpendicularAt

Function **GetPerpendicularAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

Segment

Description

The **GetPerpendicularAt** method finds the perpendicular line to a point on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A perpendicular line runs at a 90 angle to a point on a segment.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you find the perpendicular line to a point on a segment with the **GetPerpendicularAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the perpendicular line to the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetPerpendicularAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetPerpendicularAt')} **Related Topics**

Segment.GetTangentAt

Function **GetTangentAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

Segment

Description

The **GetTangentAt** method finds the tangent line to a point on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you find the tangent line to a point on a segment with the **GetTangentAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the tangent line to the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetTangentAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetTangentAt')} **Related Topics**

Segment.GetIntersections

Function **GetIntersections**(ByRef **Target** AS **Segment**, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **CrossPoints**

Segment

Description

The **GetIntersections** method finds the intersection point or points of two segments in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

In order to find an intersection with the **GetIntersections** method, you must specify the **Target** parameters:

Parameters	Description
Target	The **Target** parameter specifies the object you select last in an intersect operation. To intersect objects, you must select a first object, then select a second object that will be the target of the operation. The second selected object is called the target object. You can also intersect multiple objects with multiple target objects.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example creates a new line in the active document with the **CreateLineSegment** method and creates a new subpath of the line with the **CreateSubPath** method. The subpath is not appended to the original line, leaving the lines to be treated as two separate objects. The **GetIntersections** method determines the intersection point where to two lines meet and the lines are broken at that crosspoint. That crosspoint now serves as the first node of the detached line segment that is offset, or moved slightly, from the other line segment:

```
Sub CrossPointOffset()
Dim s As Shape
Dim spath As SubPath
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s = ActiveLayer.CreateLineSegment(0, 0, 4, 4)
Set spath = s.Curve.CreateSubPath(0, 4)
    spath.AppendLineSegment False, 4, 0
Set cps = s.Curve.Segments(1).GetIntersections(s.Curve.Segments(2))
    s.Curve.Subpaths(1).BreakApartAt cps(1).Offset
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetIntersections')} **Related Topics**

Segment.Next

Function **Next**() AS **Segment**

Segment

Description

The **Next** method returns the segment in the **Segments** collection that follows the selected segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

For example, if the selected segment has an index value of 2, the **Next** method returns the segment with an index value of 3 in the **Subpaths** collection.

Example

The following code example changes the segment type of the third segment to a line segment, if the third segment type is a curve segment and the second segment type is a curve segment:

```
Sub Test()
    Dim s As Shape
    Dim seg as Segment
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        Set seg = s.Curve.Nodes(2)
        If seg.Type = cdrCurveSegment And seg.Next.Type = cdrCurveSegment Then
            seg.Next.Type = cdrLineSegment
        End If
    End If
End Sub
```

{button ,AL(`CLS_Segment;FNC_Next')} **Related Topics**

Segment.Previous

Function **Previous**() AS **Segment**

Segment

Description

The **Previous** method returns the segment in the **Segments** collection that precedes the selected segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

For example, if the selected segment has an index value of 2, the **Next** method returns the segment with an index value of 1 in the **Subpaths** collection.

Example

The following code example changes the segment type of the first segment to a line segment, if the third segment type is a curve segment and the second segment type is a curve segment:

```
Sub Test()
    Dim s As Shape
    Dim seg as Segment
    Set s = ActiveShape
    If s.Type = cdrCurveShape Then
        Set seg = s.Curve.Nodes(2)
        If seg.Type = cdrCurveSegment And seg.Previous.Type = cdrCurveSegment Then
            seg.Previous.Type = cdrLineSegment
        End If
    End If
End Sub
```

{button ,AL(`CLS_Segment;FNC_Previous')} **Related Topics**

Segment.StartNode

Property **StartNode** AS **Node**

Segment

Description

The **StartNode** property returns the first node in a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

The **StartNode** property returns a Read-Only value.

Example

The following code example displays the horizontal coordinate of the start node in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .StartNode.PositionX
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_StartNode')} **Related Topics**

Segment.EndNode

Property **EndNode** AS **Node**

Segment

Description

The **EndNode** property returns the last node in a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

The **EndNode** property returns a Read-Only value.

Example

The following code example displays the horizontal coordinate of the end node in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    MsgBox .EndNode.PositionX
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_EndNode')} **Related Topics**

Segment.StartingControlPointX

Property **StartingControlPointX** AS Double

Segment

Description

The **StartingControlPointX** property returns or sets the horizontal coordinate of the starting control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

This property is measured in document units.

Example

The following code example sets x and y coordinates of the starting and ending control points for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointX = 2
    .StartingControlPointY = 2
    .EndingControlPointX = 5
    .EndingControlPointY = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_StartingControlPointX')} **Related Topics**

Segment.StartingControlPointY

Property **StartingControlPointY** AS Double

Segment

Description

The **StartingControlPointY** property returns or sets the vertical coordinate of the starting control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

This property is measured in document units.

Example

The following code example sets x and y coordinates of the starting and ending control points for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointX = 2
    .StartingControlPointY = 2
    .EndingControlPointX = 5
    .EndingControlPointY = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_StartingControlPointY')} **Related Topics**

Segment.EndingControlPointX

Property **EndingControlPointX** AS Double

Segment

Description

The **EndingControlPointX** property returns or sets the horizontal coordinate of the ending control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

This property is measured in document units.

Example

The following code example sets x and y coordinates of the starting and ending control points for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointX = 2
    .StartingControlPointY = 2
    .EndingControlPointX = 5
    .EndingControlPointY = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_EndingControlPointX')} **Related Topics**

Segment.EndingControlPointY

Property **EndingControlPointY** AS Double

Segment

Description

The **EndingControlPointY** property returns or sets the vertical coordinate of the ending control pointt of a segment's starting node in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

This property is measured in document units.

Example

The following code example sets x and y coordinates of the starting and ending control points for the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .StartingControlPointX = 2
    .StartingControlPointY = 2
    .EndingControlPointX = 5
    .EndingControlPointY = 5
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_EndingControlPointY')} **Related Topics**

Segment.GetPeaks

Function **GetPeaks**(ByVal **Angle** AS Double, ByRef **Offset1** AS Double, ByRef **Offset2** AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Long

Segment

Description

The **GetPeaks** method returns the peaks of a segment in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the get the peaks of a segment with the **GetCurvatureAt** method, you have the option of passing several values as parameters:

Parameters	Description
Angle	The **Angle** parameter returns the angle, measured in degrees, of the peak in a segment.
Offset1	The **Offset** parameter returns the offset distance, in document units, that an object's first node is moved out of line from another object in CorelDRAW. A duplicate object (clone) is offset from its original object (parent) when it is created - it is moved a short distance from the parent object. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object.
Offset2	The **Offset** parameter returns the offset distance, in document units, that an object's first node is moved out of line from another object in CorelDRAW. A duplicate object (clone) is offset from its original object (parent) when it is created - it is moved a short distance from the parent object. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the peak in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetPeaks (20, 1, 1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetPeaks')} **Related Topics**

Segment.GetBendPoints

Function **GetBendPoints**(ByRef **Offset1** AS Double, ByRef **Offset2** AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Long

Segment

Description

The **GetBendPoints** method gets the bend points of a segment in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the get the bend points of a segment with the **GetCurvatureAt** method, you have the option of passing several values as parameters:

Parameters	Description
Offset1	The **Offset** parameter returns the offset distance, in document units, that an object's first node is moved out of line from another object in CorelDRAW. A duplicate object (clone) is offset from its original object (parent) when it is created - it is moved a short distance from the parent object. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object.
Offset2	The **Offset** parameter returns the offset distance, in document units, that an object's first node is moved out of line from another object in CorelDRAW. A duplicate object (clone) is offset from its original object (parent) when it is created - it is moved a short distance from the parent object. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the bend point in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetBendPoints (1, 1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetBendPoints')} **Related Topics**

Segment.GetCurvatureAt

Function **GetCurvatureAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

Segment

Description

The **GetCurvatureAt** method returns a numerical value that indicates the curve ratio of a segment object in CorelDRAW. An object's curvature dictates the shape of a curve in a segment. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the curvature of a segment with the **GetCurvatureAt** method, you have the option of passing the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the curvature in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetCurvatureAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetCurvatureAt')} **Related Topics**

Segment.GetCurveSpeedAt

Function **GetCurveSpeedAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

Segment

Description

The **GetCurveSpeedAt** method returns a numerical value that indicates the curve speed of a segment object in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the speed of a segment with the **GetCurveSpeedAt** method, you have the option of passing the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the curve speed in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetCurveSpeedAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetCurveSpeedAt')} **Related Topics**

Segment.FindParamOffset

Function **FindParamOffset**(ByVal **AbsoluteOffset** As Double, ByRef **ParamOffset** As Double, [ByRef Remainder As Double]) As Boolean

Description

The **FindParamOffset** method returns or sets a True or False value that indicates the parametric offset of a segment in COrelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the parametric offset of a segment with the **FindParamOffset** method, you have the option of passing several values as parameters:

Parameters	Description
AbsoluteOffset	Lets you specify the absolute offset value.
ParamOffset	Lets you specify the parametric offset value.
Remainder	Lets you specify the Remainder value. This value is optional.

Example

The following code example gets the parametric offset in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .FindParamOffset (1, 1)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_FindParamOffset')} **Related Topics**

Segment.GetAbsoluteOffset

Function **GetAbsoluteOffset**(ByVal **ParamOffset** As Double) As Double

Description

The **GetAbsoluteOffset** method returns or sets a True or False value that indicates the parametric offset of a segment in COrelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the absolute offset of a segment with the **GetAbsoluteOffset** method, you must pass the **ParamOffset** value as parameters:

Parameters	Description
ParamOffset	The **ParamOffset** parameter is a numeric value that lets you specify the parametric offset value of a segment.

Example

The following code example gets the absolute offset in the first segment in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Segments(1)
    .GetAbsoluteOffset (2)
End With
End Sub
```

{button ,AL(`CLS_Segment;FNC_GetAbsoluteOffset')} **Related Topics**

Segments properties

Segments Legend

- Application
- Count
-
- Item
- Parent

Segments methods

Segments Legend

- All
- AllExcluding
- Range

The square points at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes.

Segments

Class **Segments**

Properties Methods Referenced by

The **Segments** class defines the characteristics of **Segments** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

You can also change the direction of a curve and smooth out a curve using buttons on the Shape tool Property Bar.

{button ,AL(`CLS_Segments')} **Related Topics**

Segments.Application

Property **Application** AS **Application**

Segments

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub SegmentsApp()
With ActiveShape.Curve.Segments
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_Application')} **Related Topics**

Segments.Parent

Property **Parent** AS **Shape**

Segments

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the segment's parent object in a message box. The length of the curve object is measured in document units:

```
Sub SegmentsParent()
With ActiveShape.Curve.Segments
    MsgBox .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_Parent')} **Related Topics**

Segments.Item

Property **Item**(ByVal **Index** AS Long) AS **Segment**

Segments

Description

The **Item** property returns a value associated with the index number of a Segment object in the **Segments** collection of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

`Segments(2)` refers to the second segment object in the **Segments** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Segments.Item(2)` is the same as `Segments(2)` - they both reference the second segment object in the **Segments** collection.

You must reference a segment in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Segments** collection; it uniquely identifies each member of the collection.

Example

The following code example displays a number associated with the segment type of the first segment in the selected curve object of CorelDRAW:

```
Sub SegmentsItem()
With ActiveShape.Curve
    MsgBox .Segments(1).Type
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_Item')} **Related Topics**

cdrLineSegment=0

cdrCurveSegment=1

cdrMixedSegments=2

Segments.Count

Property **Count** AS Long

Segments

Description

The **Count** property returns the number of segments in the **Segments** collection of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of segments in the selected curve object in CorelDRAW:

```
Sub SegmentsCount()
With ActiveShape.Curve.Segments
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_Count')} **Related Topics**

Segments.Range

Function **Range**(ByVal **IndexArray** AS Variant) AS **SegmentRange**

Segments

Description

The **Range** method returns or sets a value associated with a series of segment objects in the **Segments** collection in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **Range** method allows you to reference several segments concurrently. The use of an index range makes it easier and faster to find, sort, and edit multiple segments in the **Segments** collection.

You must reference the segments in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify specific segments in the **Segments** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific segment objects that become members of the array.

Example

The following code example adds a node to each member of a segment array. Two segments in the selected curve object, the first and the third, are included in the array and a node is added to each segment. A message box displays the number of nodes in the curve object before and after the nodes are added so you can see the addition of the two nodes to the curve object:

```
Sub SegmentsRange()
MsgBox ActiveShape.Curve.Nodes.Count
With ActiveShape.Curve.Segments
    .Range(Array(1,3)).AddNode
End With
MsgBox ActiveShape.Curve.Nodes.Count
```

End Sub

{button ,AL(`CLS_Segments;FNC_Range')} **Related Topics**

Segments.All

Function **All**() AS **SegmentRange**

Segments

Description

The **All** method returns or sets a value associated with all segment objects in the **Segments** collection in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **All** method allows you to reference all segments concurrently. The use of the **All** method makes it easier and faster to find, sort, and edit all segments in the **Segments** collection.

Example

The following code example adds a node to all segments in the segments collection of the selected curve object in CorelDRAW. A message box displays the number of nodes before and after the nodes are added so you can see that the nodes have been added to the curve object:

```
Sub SegmentsAll()
With ActiveShape.Curve.Segments
MsgBox ActiveShape.Curve.Nodes.Count
.All.AddNode
MsgBox ActiveShape.Curve.Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_All')} **Related Topics**

Segments.AllExcluding

Function **AllExcluding**(ByVal **IndexArray** AS Variant) AS **SegmentRange**

Segments

Description

The **AllExcluding** method returns or sets a value associated with a series of segment objects in the **Segments** collection in CorelDRAW, with the exception of the segments identified by their index numbers in the method parameters. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

You must reference the segments in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify and group specific segments in the **Segments** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific segment objects that become members of the array.

Example

The following code example adds a node to all segments in the segments collection of the selected curve object, except for the segments identified the array of the **AllExcluding** method. A message box displays the number of nodes before and after the nodes are added so you can see that the nodes have been added to the curve object:

```
Sub SegmentsAllEx()
With ActiveShape.Curve.Segments
MsgBox ActiveShape.Curve.Nodes.Count
.AllExcluding (Array(1,6)).AddNode
MsgBox ActiveShape.Curve.Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_Segments;FNC_AllExcluding')} **Related Topics**

SegmentRange properties

SegmentRange Legend

- Application
- Count
-
- Item
- Length
- NodeRange
- Parent
- Type

SegmentRange methods

SegmentRange Legend

- Add
- AddNode
- AddRange
- Remove
- RemoveAll
- SetType

SegmentRange

Class **SegmentRange**

Properties Methods Referenced by

The **SegmentRange** class defines the characteristics of a range of segment objects and describes the look and behavior of the object range through its properties and methods.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

Once you have defined a range, you can perform operations on it. This is a powerful feature because it allows you to manipulate the components of the range with one expression.

{button ,AL(`CLS_SegmentRange')} **Related Topics**

A range is series of similar objects in CorelDRAW.

In programming, an expression is any legal combination of symbols that represents a value.

SegmentRange.NodeRange

Property **NodeRange** As **NodeRange**

Description

The **NodeRange** property returns a value associated with a range of nodes that correspond to a segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

The **NodeRange** property returns a Read-Only value.

Example

The following code example creates two separate segment ranges from the active shape, using the **Range** and **AllExluding** methods. Both segment ranges are combined using the **AddRange** method. The number of nodes in displayed in a message box:

```
Sub Range()
Dim s as SegmentRange
Dim NewRange as New SegmentRange
Set NewRange = ActiveShape.Curve.Segments.Range (1, 2)
'creates a range with the first and second segment in the shape
Set s = ActiveShape.Curve.Segments.AllExcluding (1, 2)
'creates a range with all segments except the first and second segments
With s
    .AddRange NewRange
    .MsgBox .NodeRange.Count
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_NodeRange')} **Related Topics**

SegmentRange.AddRange

Sub **AddRange**(ByVal **SegmentRange** As **SegmentRange**)

Description

The **AddRange** method adds a segment range to an existing segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

In order the use the **AddRange** method, you must pass the **SegmentRange** value as a parameter:

Parameters	Description
SegmentRange	The **SegmentRange** parameter specifies the range of segment items that are added to current segment range. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

Example

The following code example creates two separate segment ranges from the active shape, using the **Range** and **AllExluding** methods. Both segment ranges are combined using the **AddRange** method. The number of nodes in displayed in a message box:

```
Sub Range()
Dim s as SegmentRange
Dim NewRange as New SegmentRange
Set NewRange = ActiveShape.Curve.Segments.Range (1, 2)
'creates a range with the first and second segment in the shape
Set s = ActiveShape.Curve.Segments.AllExcluding (1, 2)
'creates a range with all segments except the first and second segments
With s
    .AddRange NewRange
    .MsgBox .NodeRange.Count
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_AddRange')} **Related Topics**

SegmentRange.RemoveAll

Sub **RemoveAll**()

Description

The **RemoveAll** method removes all segments from a segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

Example

The following code example makes all the segments in the active shape line segments, using the **SetType** method. All the segments are then deleted with the **RemoveAll** method:

```
Sub SegType()
Dim s as SegmentRange
Set s = ActiveShape.Curve.Segments.All
'creates a range with all segments in the shape
With s
    .SetType cdrlineSegment
    .RemoveAll
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_RemoveAll')} **Related Topics**

SegmentRange.SetType

Sub **SetType**(ByVal **Type** As **cdrSegmentType**)

Description

The **SetType** method sets the type of all segments within a segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **SetType** method returns a value of cdrSegmentType.

In order to use the **SetType** method, you must pass the **Type** value as a parameter:

Parameters	Description
Type	The **Type** parameter specifies the segment type for all the segments within a segment range. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. The **Type** value returns cdrSegmentType.

Example

The following code example makes all the segments in the active shape line segments, using the **SetType** method:

```
Sub SegType()
Dim s as SegmentRange
Set s = ActiveShape.Curve.Segments.All
'creates a range with all segments in the shape
With s
    .SetType cdrlineSegment
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_SetType')} **Related Topics**

SegmentRange.Application

Property **Application** AS **Application**

SegmentRange

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW is a message box:

```
Sub SegRangeApp()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
With segrange
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Application')} **Related Topics**

SegmentRange.Parent

Property **Parent** AS **Shape**

SegmentRange

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the parent curve object in a message box:

```
Sub SegRangePar()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Semgents.All
With segrange
    MsgBox "The length of the parent curve object is " & .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Parent')} **Related Topics**

SegmentRange.Item

Property **Item**(ByVal **Index** AS Long) AS **Segment**

SegmentRange

Description

The **Item** property returns a value associated with the index number of a SegmentRange object in the **SegmentRange** class of CorelDRAW. A segment is a line or curve between nodes in a curve object.There are two types of segments: straight and curved.

`SegmentRange(2)` refers to the second segment range object in the **SegmentRange** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the range. For example, `SegmentRange.Item(2)` is the same as `SegmentRange(2)` - they both reference the second segment range object in the range.

You must reference a member of a segment range by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **SegmentsRange** class: it uniquely identifies each member of the segment range.

Example

The following code example displays the length, in document units, of the first segment in the segment range in a message box:

```
Sub SegRangeItem()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
With segrange
    MsgBox .Item(1).Length
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Item')} **Related Topics**

SegmentRange.Count

Property **Count** AS Long

SegmentRange

Description

The **Count** property returns the number of segments in the **SegmentRange** class of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of segments in the segment range in a message box:

```
Sub SegRangeCount()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
With segrange
    MsgBox "There are " & .Count & " segments in the range."
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Count')} **Related Topics**

SegmentRange.Type

Property **Type** AS **cdrSegmentType**

SegmentRange

Description

The **Type** property returns a value associated with a segment type in the **SegmentRange** class of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

If you are using the **Type** property for the entire segment range, a value is returned only if the segment type is common to all segments included within the segment range. The **Type** property can also be used with the Item property to return the segment type for a specific segment within the segment range.

The **Type** property returns a value of cdrSegmentType.

Example

The following code example displays the segment type of the second segment in the segment range:

```
Sub SegRangeType()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
With segrange
    MsgBox .Item(2).Type
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Type')} **Related Topics**

SegmentRange.Length

Property **Length** AS Double

SegmentRange

Description

The **Length** property returns a value associated with the total length of all segments within a range in the **SegmentRange** class of CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The length is measured in document units.

The **Length** property returns a Read-Only value.

Example

The following code example displays the length, in document units, of the first segment in the segment range of the selected curve object:

```
Sub SegRangeLength()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
With segrange
    MsgBox Item(1).Length
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Length')} **Related Topics**

SegmentRange.Add

Sub **Add**(ByRef **Segment** AS **Segment**)

SegmentRange

Description

The **Add** method places a specified segment object into an existing segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

You must reference a member of a segment range by passing the segment object to be added as a parameter:

Parameters	Description
Segment	The **Segment** parameter specifies the segment object in the **Segments** collection that is added to the segment range. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

Example

The following code example adds the first segment in the **Segments** collection to the segment range, using the **Add** method:

```
Sub SegAdd()
Dim sr as SegmentRange
Set sr = ActiveShape.Curve.Segments.AllExcluding (1)
With sr
    .Add ActiveShape.Curve.Segments (1)
End With
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Add')} **Related Topics**

SegmentRange.Remove

Sub **Remove**(ByVal **Index** AS Long)

SegmentRange

Description

The **Remove** method takes a specified segment object out of an existing segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

You must reference a member of a segment range by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **SegmentsRange** class: it uniquely identifies each member of the segment range.

Example

The following code example removes the first segment (the segment with the index value of 1) from the segment range. A message box displays the number of segments in the range before and after the Remove method is executed, so you can see that the segment was removed from the range:

```
Sub SegRangeRemove()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
MsgBox s.Count
With segrange
    .Remove
End With
MsgBox s.Count
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_Remove')} **Related Topics**

SegmentRange.AddNode

Sub **AddNode**()

SegmentRange

Description

The **AddNode** method adds a node at the middle of each segment object in a segment range in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

Nodes are added to curve objects to change the shape and appearance of the object in CorelDRAW.

Example

The following code example adds a node to each segment in the segment range. A message box displays the number of nodes before and after the **AddNode** method is executed, so you can see that the nodes have been added to the segments:

```
Sub SegRangeAddNode()
Dim segrange as SegmentRange
Set segrange = ActiveShape.Curve.Segments.All
MsgBox s.Parent.Curve.Nodes.Count
With segrange
    .AddNode
End With
MsgBox s.Parent.Curve.Nodes.Count
End Sub
```

{button ,AL(`CLS_SegmentRange;FNC_AddNode')} **Related Topics**

SubPath properties

SubPath Legend

- Application
  - Closed
- EndNode
- FirstSegment
- Index
- LastSegment
- Length
- Nodes
- Parent
  - PositionX
  - PositionY
- Segments
- SizeHeight
- SizeWidth
- StartNode

SubPath methods

SubPath Legend

- AddNodeAt
- AppendCurveSegment
- AppendLineSegment
- BreakApartAt
- Delete
- Extract
- FindSegmentOffset
- GetCurvatureAt
- GetCurveSpeedAt
- GetIntersections
- GetPerpendicularAt
- GetPointPositionAt
- GetPosition
- GetSegmentAt
- GetTangentAt
- IsOnSubPath
- Move
- Next
- Previous
- ReverseDirection
- Selection
- SetPosition

SubPath

Class **SubPath**

Properties Methods Referenced by

The **Subpath** class defines the characteristics of subpath objects and describes the look and behavior of the collection objects through its properties and methods.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

You can turn a closed curve object into an open one by breaking its path at any point. You can also break an open path into one or more subpaths or into separate objects.

When you break a path, any subpaths and nodes that are created remain a part of the original object.

{button ,AL(`CLS_SubPath')} **Related Topics**

SubPath.FirstSegment

Property **FirstSegment** As **Segment**

Description

The **FirstSegment** property returns the first segment in a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **FirstSegment** property returns a Read-Only value.

Example

The following code example displays the length of the first segment in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .FirstSegment.Length
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_FirstSegment')} **Related Topics**

SubPath.LastSegment

Property **LastSegment** As **Segment**

Description

The **LastSegment** property returns the last segment in a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved.

The **LastSegment** property returns a Read-Only value.

Example

The following code example displays the length of the last segment in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .LastSegment.Length
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_LastSegment')} **Related Topics**

SubPath.Application

Property **Application** AS **Application**

SubPath

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Application')} **Related Topics**

SubPath.Parent

Property **Parent** AS **Shape**

SubPath

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the first subpath's parent curve in CorelDRAW::

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Parent')} **Related Topics**

SubPath.Nodes

Property **Nodes** AS **Nodes**

SubPath

Description

The **Nodes** property returns a value that defines the characteristics of a subpath's **Nodes** collection object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the node type.

The **Nodes** property returns a Read-Only value.

Example

The following code example counts the number of nodes in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Nodes')} **Related Topics**

SubPath.Segments

Property **Segments** AS **Segments**

SubPath

Description

The **Segments** property returns a value that defines the characteristics of a subpath's **Segments** collection object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

You can also change the direction of a curve and smooth out a curve using buttons on the Shape tool Property Bar.

The **Segments** property returns a Read-Only value.

Example

The following code example counts the number of segments in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Segments.Count
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Segments')} **Related Topics**

SubPath.Index

Property **Index** AS Long

SubPath

Description

The **Index** property returns the index number of a curve's subpath in the **Subpaths** collection of CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

Each subpath in the **Subpaths** collection is uniquely identified by an index number. For example, Subpaths(5) refers to the fifth subpath in the **Subpaths** collection of CorelDRAW.

The **Index** property returns a Read-Only value.

Example

The following code example uses the index value of 1 to identify the first subpath in the active shape. The length of the first subpath in the active shape, in document units, displays in a message box:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Length
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Index')} **Related Topics**

SubPath.Length

Property **Length** AS Double

SubPath

Description

The **Length** property returns a value associated with the total length of a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

The **Length** property is measured in document units.

The **Length** property returns a Read-Only value.

Example

The following code example uses the index value of 1 to identify the first subpath in the active shape. The length of the first subpath in the active shape, in document units, displays in a message box:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Length
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Length')} **Related Topics**

SubPath.Closed

Property **Closed** AS Boolean

SubPath

Description

The **Closed** property returns or sets a True or False value that indicates if a curve's subpath is opened or closed in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

A closed path is a path that encloses an area because the path's start and end points are connected. If the **Closed** property is True, the subpath is closed.

Example

The following code example closes the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .Closed = True
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Closed')} **Related Topics**

SubPath.PositionX

Property **PositionX** AS Double

SubPath

Description

The **PositionX** property returns or sets a numerical value that indicates the horizontal placement of a subpath's head in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **PositionX** property returns or sets the X coordinate of a subpath's start point in CorelDRAW.

Example

The following code example sets the x and y coordinates of the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .PositionX = 2
    .PositionY = 2
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_PositionX')} **Related Topics**

SubPath.PositionY

Property **PositionY** AS Double

SubPath

Description

The **PositionY** property returns or sets a numerical value that indicates the vertical placement of a subpath's head in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **PositionY** property returns or sets the Y coordinate of a subpath's start point in CorelDRAW.

Example

The following code example sets the x and y coordinates of the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .PositionX = 2
    .PositionY = 2
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_PositionY')} **Related Topics**

SubPath.SizeWidth

Property **SizeWidth** AS Double

SubPath

Description

The **SizeWidth** property returns a numerical value that indicates the width of an entire subpath object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **SizeWidth** property is measured in document units.

The **SizeWidth** property returns a Read-only value.

Example

The following code example displays the width and height, in document units, of the first subpath in the active shape of CoreDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .SizeWidth
    MsgBox .SizeHeight
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_SizeWidth')} **Related Topics**

SubPath.SizeHeight

Property **SizeHeight** AS Double

SubPath

Description

The **SizeHeight** property returns a numerical value that indicates the height of an entire subpath object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **SizeHeight** property is measured in document units.

The **SizeHeight** property returns a Read-only value.

Example

The following code example displays the width and height, in document units, of the first subpath in the active shape of CoreDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .SizeWidth
    MsgBox .SizeHeight
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_SizeHeight')} **Related Topics**

SubPath.Selection

Function **Selection**() AS **NodeRange**

SubPath

Description

The **Selection** method creates a range of nodes on a subpath in CorelDRAW. This node range contains all the selected nodes on the subpath. A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the node type.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **Selection** method creates a node range and allows you to reference several nodes concurrently. The use of an index range makes it easier and faster to find, sort, and edit multiple nodes in the **Nodes** collection.

Example

The following code example counts the number of nodes in the node selection of the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .Selection.Count
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Selection')} **Related Topics**

SubPath.ReverseDirection

Sub **ReverseDirection**()

SubPath

Description

The **ReverseDirection** method reverses the node order on a subpath in CorelDRAW. Each node on a subpath has an index number. The **ReverseDirection** method reverses the order of the node's index numbers. For example, the first node in a **Nodes** collection becomes the last node in the collection when the **ReverseDirection** method is invoked in CorelDRAW.

A node is the square point at the end of lines and curve segments. You can change the shape of a line or curve by dragging one or more of its nodes. You can add nodes to a curve object if you need more nodes to create the curve you want. You must select a node or a segment before you can make changes to the curve.

When you select a node attached to a straight segment, the node appears as an unfilled square. If the node is attached to a curve segment, it appears as a solid square. The Status Bar indicates the node type.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

Example

The following code example reverses the node direction in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .ReverseDirection
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_ReverseDirection')} **Related Topics**

SubPath.Extract

Function **Extract**(ByRef OldCurve AS **Shape**) AS **Shape**

SubPath

Description

The **Extract** method removes a subpath from an existing curve object and creates a new curve from the extracted subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

When you break a path in an object it still remains as one object. When you extract a subpath, you create two separate objects.

In order to extract a subpath with the **Extract** method, you must reference the original curve object as the **OldCurve** parameter:

Parameters	Description
OldCurve	The **OldCurve** parameter references the original curve object containing the subpath that is extracted and made into a separate curve object in CorelDRAW. The **OldCurve** parameter is a shape object and the parameter is optional - if no shape is referenced, the active shape object is passed **OldCurve** parameter.

{button ,AL(`CLS_SubPath;FNC_Extract')} **Related Topics**

SubPath.Delete

Sub **Delete**()

SubPath

Description

The **Delete** method removes an entire subpath from a curve object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

Example

The following code example deletes the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .Delete
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Delete')} **Related Topics**

SubPath.GetPosition

Sub **GetPosition**(ByRef **PositionX** AS Double, ByRef **PositionY** AS Double)

SubPath

Description

The **GetPosition** method returns the horizontal and vertical position of a subpaths' head in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to return the position of the subpath's head with the **GetPosition** method, you must pass the X and Y positions of the subpath as parameters:

Parameters	Description
PositionX	The **PositionX** parameter identifies the **X**, or horizontal, coordinate of a subpath's start point in CorelDRAW. The **X** coordinate is measured in document units.
PositionY	The **PositionY** parameter identifies the **Y**, or vertical, coordinate of a subpath's start point in CorelDRAW. The **Y** coordinate is measured in document units.

Example

The following code example gets the horizontal and vertical position of the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetPosition (4, 4)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetPosition')} **Related Topics**

SubPath.SetPosition

Sub **SetPosition**(ByVal **PositionX** AS Double, ByVal **PositionY** AS Double)

SubPath

Description

The **SetPosition** method sets the horizontal and vertical position of a subpaths' head in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to set the position of the subpath's head with the **SetPosition** method, you must pass the X and Y positions of the subpath as parameters:

Parameters	Description
PositionX	The **PositionX** parameter identifies the **X**, or horizontal, coordinate of a subpath's start point in CorelDRAW. The **X** coordinate is measured in document units.
PositionY	The **PositionY** parameter identifies the **Y**, or vertical, coordinate of a subpath's start point in CorelDRAW. The **Y** coordinate is measured in document units.

Example

The following code example sets the horizontal and vertical position of the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .SetPosition (4, 4)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_SetPosition')} **Related Topics**

SubPath.Move

Sub **Move**(ByVal DeltaX AS Double, ByVal DeltaY AS Double)

SubPath

Description

The **Move** method changes the position of a curve's entire subpath by moving it a specified distance from its original location in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to move a subpath with the **Move** method, you must pass the **DeltaX** and **DeltaY** values as parameters:

Parameters	Description
DeltaX	The **DeltaX** parameter specifies the horizontal distance to move a subpath. Delta refers to a limited incremental value in a variable. The default value is 0.
DeltaY	The **DeltaY** parameter specifies the vertical distance to move a subpath. Delta refers to a limited incremental value in a variable. The default value is 0.

Example

The following code example moves the first subpath in the active shape over and up 1:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .Move 1, 1
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Move')} **Related Topics**

SubPath.IsOnSubPath

Function **IsOnSubPath**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal HotArea AS Double) AS **cdrPositionOfPointOverShape**

SubPath

Description

The **IsOnSubPath** method specifies the position of a coordinate in relation to a curve's subpath in CorelDRAW. A coordinate can be inside, outside, or on the margin of a subpath object.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **IsOnSubPath** method returns a value of cdrPositionOfPointOverShape.

In order to specify the location of a coordinate with the **IsOnSubPath** method, you must pass the **X** and **Y** coordinates of the point, or a hot spot area as parameters. The **HotSpot** parameter limits the area to search in the subpath:

Parameters	Description
X	The **X** parameter specifies the horizontal position (using the rulers as a reference) of a point in CorelDRAW. The **IsOnSubPath** method determines if the point is on a curve's subpath. The **X** coordinate is measured in document units.
Y	The **Y** parameter specifies the vertical position (using the rulers as a reference) of a point in CorelDRAW. The **IsOnSubPath** method determines if the point is on a curve's subpath. The **Y** coordinate is measured in document units.
HotArea	The HotArea parameter defines a restricted area on the curve's subpath in which to evaluate the position of a point, in relation to the curve. This value is optional.

Example

The following code example returns the position of point (2, 3) in relation to the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .IsOnSubPath (2, 3)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_IsOnSubPath')} **Related Topics**

SubPath.AppendLineSegment

Function **AppendLineSegment**(ByVal AppendAtBeginning AS Boolean, ByVal **X** AS Double, ByVal **Y** AS Double) AS **Segment**

SubPath

Description

The **AppendLineSegment** method joins a new line segment to an existing curve's subpath in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself. A line segment cannot be bent. In order to change the shape of a line segment, it must be converted to a curve segment.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to append a new line segment to a subpath with the **AppendLineSegment** method, you must pass the **X**, **Y**, and **AppendAtBeginning** values as parameters:

Parameters	Description
AppendAtBeginning	The **AppendAtBeginning** parameter is a True or False value that specifies if the new line segment joins the subpath at the head of the subpath object. This value is optional and the default value is False.
X	The **X** parameter specifies the horizontal position of the new line segment that is appended to the subpath object. The X coordinate is measured in document units.
Y	The **Y** parameter specifies the vertical position of the new line segment that is appended to the subpath object. The Y coordinate is measured in document units.

Example

The following code example creates a new curve object in the active document and creates a new subpath in that curve object. Using the coordinates (1, 1), the new subpath appears one inch to the right and one inch above the bottom left corner of the active page, set by the reference point.. Within that new subpath, two un-appended line segments are added with the **AppendLineSegment** method. The subpath is closed to create an inverted closed triangle:

```
Sub SubPathCreate()
Dim s As Shape
Dim sp As SubPath
Set s = ActiveLayer.CreateCurve
ActiveDocument.ReferencePoint = cdrBottomLeft
Set sp = s.Curve.CreateSubPath(1, 1)
    sp.AppendLineSegment False, 2, 2
    sp.AppendLineSegment False, 0, 2
    sp.Closed = True
End Sub
```

{button ,AL(`CLS_SubPath;FNC_AppendLineSegment')} **Related Topics**

SubPath.AppendCurveSegment

Function **AppendCurveSegment**(ByVal AppendAtBeginning AS Boolean, ByVal **X** AS Double, ByVal **Y** AS Double, ByVal StartingControlPointLength AS Double, ByVal StartingControlPointAngle AS Double, ByVal EndingControlPointLength AS Double, ByVal EndingControlPointAngle AS Double) AS **Segment**

SubPath

Description

The **AppendCurveSegment** method joins a new curve segment to an existing curve's subpath in CorelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. A line segment cannot be bent. In order to change the shape of a line segment, it must be converted to a curve segment.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to append a new curve segment to a subpath with the **AppendCurveSegment** method, you must pass several parameters:

Parameters	Description
AppendAtBeginning	The **AppendAtBeginning** parameter is a True or False value that specifies if the new line segment joins the subpath at the head of the subpath object. This value is optional and the default value is False.
X	The **X** parameter specifies the horizontal position of the new line segment that is appended to the subpath object. The **X** coordinate is measured in document units.
Y	The **Y** parameter specifies the vertical position of the new line segment that is appended to the subpath object. The **Y** coordinate is measured in document units.
StartingControlPointLength	The StartingControlPointLength parameter specifies the distance of the starting control pointControlPoint_definition from the node that lies between the new appended curve segment and the existing curve's subpath object. The length is measured in document units. The length and angle of the control point dictate the appearance of the appended segment object. This value is optional and the default value is -1.
StartingControlPointAngle	The **StartControlPointAngle** parameter specifies the angle, measured in degrees, of the starting control point. You can control the curve of a segment by varying the control point's angle and its distance from the node. This value is optional and the default value is 0.
EndingControlPointLength	The **StartingControlPointLength** parameter specifies the distance of the starting control point from the node that lies between the new appended curve segment and the existing curve's subpath object. The length is measured in document units. The length and angle of the control point dictate the appearance of the appended segment object. This value is optional and the default value is -1.
EndingControlPointAngle	The **EndControlPointAngle** parameter specifies the angle, measured in degrees, of the ending control point. You can control the curve of a segment by varying the control point's angle and its distance from the node. This value is optional and the default value is 0.

{button ,AL(`CLS_SubPath;FNC_AppendCurveSegment')} **Related Topics**

SubPath.GetPointPositionAt

Sub **GetPointPositionAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**, ByRef **X** AS Double, ByRef **Y** AS Double)

SubPath

Description

The **GetPointPositionAt** method returns the coordinates of a point located on a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to return a point position with the **GetPointPositionAt** method, you must pass the **Offset**, **OffsetType**, **X** and **Y** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a curve's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).
X	The **X** parameter specifies the horizontal position of a point on a curve's subpath. The **X** coordinate is measured in document units.
Y	The **Y** parameter specifies the vertical position of a point on a curve's subpath. The **Y** coordinate is measured in document units.

Example

The following code example gets the point position at coordinate (2, 2) in the first subpath of the active shape in CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetPointPositionAt (1, cdrRelativeSegmentOffset, 2, 2)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetPointPositionAt')} **Related Topics**

cdrAbsoluteSegmentOffset=0

cdrRelativeSegmentOffset=1

cdrParamSegmentOffset=2

SubPath.BreakApartAt

Function **BreakApartAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **Node**

SubPath

Description

The **BreakApartAt** method separates a segment into two unconnected segments in CorelDRAW. A segment is the part of a curve lying between two nodes. Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

In order to break a segment with the **BreakApartAt** method, you must pass the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, to place the broken portion of the segment from the original segment. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of the broken portion of the segment. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example creates a new line in the active document with the **CreateLineSegment** method and creates a new subpath of the line with the **CreateSubPath** method. The subpath is not appended to the original line, leaving the lines to be treated as two separate objects. The **GetIntersections** method determines the intersection point where to two lines meet and the lines are broken at that crosspoint. That crosspoint now serves as the first node of the detached line segment that is offset, or moved slightly, from the other line segment:

```
Sub CrossPointOffset()
Dim s As Shape
Dim spath As SubPath
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s = ActiveLayer.CreateLineSegment(0, 0, 4, 4)
Set spath = s.Curve.CreateSubPath(0, 4)
    spath.AppendLineSegment False, 4, 0
Set cps = s.Curve.Segments(1).GetIntersections(s.Curve.Segments(2))
    s.Curve.Subpaths(1).BreakApartAt cps(1).Offset
End Sub
```

{button ,AL(`CLS_SubPath;FNC_BreakApartAt')} **Related Topics**

SubPath.AddNodeAt

Function **AddNodeAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **Node**

SubPath

Description

The **AddNodeAt** method inserts a node on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you add a new node to a segment with the **AddNodeAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of the new node on a segment. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of the new node on a segment. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example adds a new node to the first subpath of the active shape in CorelDRAW, offset by 1:

```
SubApp()
With ActiveShape.Curve.Subpaths(1)
    .AddNodeAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_AddNodeAt')} **Related Topics**

SubPath.GetPerpendicularAt

Function **GetPerpendicularAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

SubPath

Description

The **GetPerpendicularAt** method finds the perpendicular line to a point on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A perpendicular line runs at a 90 angle to a point on a segment.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you find the perpendicular line to a point on a segment with the **GetPerpendicularAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of the perpendicular line. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of the perpendicular line. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the perpendicular line to the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetPerpendicularAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetPerpendicularAt')} **Related Topics**

SubPath.GetTangentAt

Function **GetTangentAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

SubPath

Description

The **GetTangentAt** method finds the tangent line to a point on a segment in CorelDRAW. A segment is the part of a curve lying between two nodes. A tangent line touches the outside of a curve object without intersecting the curve.

Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

When you find the tangent line to a point on a segment with the **GetTangentAt** method, you have the option of specifying the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of the tangent line. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of the tangent line. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the tangent line to the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetTangentAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetTangentAt')} **Related Topics**

SubPath.GetIntersections

Function **GetIntersections**(ByRef **Target** AS **SubPath**, ByVal OffsetType AS **cdrSegmentOffsetType**) AS **CrossPoints**

SubPath

Description

The **GetIntersections** method finds the intersection point or points of two subpath objects in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

In order to find an intersection with the **GetIntersections** method, you must specify the **Target** parameter:

Parameters	Description
Target	The **Target** parameter specifies the object you select last in an intersect operation. To intersect objects, you must select a first object, then select a second object that will be the target of the operation. The second selected object is called the target object. You can also intersect multiple objects with multiple target objects.
OffsetType	The **OffsetType** parameter specifies the offset type of the intersection point. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

{button ,AL(`CLS_SubPath;FNC_GetIntersections')} **Related Topics**

SubPath.GetSegmentAt

Function **GetSegmentAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**, ByRef SegmentOffset AS Double) AS **Segment**

SubPath

Description

The **GetSegmentAt** method returns a segment at a specified position on a curve's subpath in CorelDRAW. A segment is the part of a curve lying between two nodes. Segments are of two types: curved or straight. You can bend a curved segment by dragging it with the Shape tool or by dragging the control points of the nodes on either end of it. If you want to bend a straight segment, you must convert it to a curved segment.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

When you find a segment at a specified position with the **GetSegmentAt** method, you have the option of passing the **Offset**, **OffsetType**, and **SegmentOffset** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of the tangent line. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of the intersection point. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).
SegmentOffset	The **SegmentOffset** parameter returns the offset value of a segment in the subpath.

{button ,AL(`CLS_SubPath;FNC_GetSegmentAt')} **Related Topics**

SubPath.Next

Function **Next**() AS **SubPath**

SubPath

Description

The **Next** method returns the subpath in the **Subpaths** collection that follows the selected subpath object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

For example, if the selected subpath object has an index value of 2, the **Next** method returns the subpath with an index value of 3 in the **Subpaths** collection.

Example

The following code example changes the node type of the end node in the second subpath of the active shape, only if the end node in the first subpath is a symmetrical node:

```
Sub Test()
    Dim s As Shape
    Dim spath as Subpath
    Set s = ActiveShape
    Set spath = s.Curve.Subpaths(1)
      If spath.EndNode.Type = cdrSymmetricalNode Then
      spath.Next.EndNode.Type = cdrSmoothNode
      End If
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Next')} **Related Topics**

SubPath.Previous

Function **Previous**() AS **SubPath**

SubPath

Description

The **Previous** method returns the subpath in the **Subpaths** collection that precedes the selected subpath object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

For example, if the selected subpath object has an index value of 2, the **Previous** method returns the subpath with an index value of 1 in the **Subpaths** collection.

Example

The following code example changes the node type of the end node in the second subpath of the active shape, only if the end node in the third subpath is a symmetrical node:

```
Sub Test()
    Dim s As Shape
    Dim spath as Subpath
    Set s = ActiveShape
    Set spath = s.Curve.Subpaths(3)
      If spath.EndNode.Type = cdrSymmetricalNode Then
      spath.Previous.EndNode.Type = cdrSmoothNode
      End If
End Sub
```

{button ,AL(`CLS_SubPath;FNC_Previous')} **Related Topics**

SubPath.GetCurvatureAt

Function **GetCurvatureAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

SubPath

Description

The **GetCurvatureAt** method returns a numerical value that indicates the curve ratio of a subpath object in CorelDRAW. An object's curvature dictates the shape of a curve in a subpath. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

When you find the curvature of a subpath with the **GetCurvatureAt** method, you have the option of passing the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units of a subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offset type of a subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the curvature in the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetCurvatureAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetCurvatureAt')} **Related Topics**

SubPath.GetCurveSpeedAt

Function **GetCurveSpeedAt**(ByVal Offset AS Double, ByVal OffsetType AS **cdrSegmentOffsetType**) AS Double

SubPath

Description

The **GetCurveSpeedAt** method returns a numerical value that indicates the curve speed of a subpath's curve object in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

When you find the speed of a curve with the **GetCurveSpeedAt** method, you have the option of passing the **Offset** and **OffsetType** values as parameters:

Parameters	Description
Offset	The **Offset** parameter specifies the offset distance, in document units, of a point on a segment's subpath. Positive values indicate an offset effect upward and to the right, while negative offset values indicate a move downward and to the left of the specified object. This value is optional and the default value of the offset distance .5.
OffsetType	The **OffsetType** parameter specifies the offsettype of a point on a curve's subpath. This parameter returns cdrSegmentOffsetType. This value is optional and the default value is cdrRelativeSegmentOffset (1).

Example

The following code example gets the curve speed in the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    .GetCurveSpeedAt (1, cdrAbsoluteSegmentOffset)
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_GetCurveSpeedAt')} **Related Topics**

SubPath.StartNode

Property **StartNode** AS **Node**

SubPath

Description

The **StartNode** property returns the first node in a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **StartNode** property returns a Read-Only value.

Example

The following code example displays the horizontal coordinate of the start node in the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .StartNode.PositionX
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_StartNode')} **Related Topics**

SubPath.EndNode

Property **EndNode** AS **Node**

SubPath

Description

The **EndNode** property returns the last node in a curve's subpath in CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. A subpath's head is the starting point of the subpath object.

The **EndNode** property returns a Read-Only value.

Example

The following code example displays the horizontal coordinate of the end node in the first subpath in the active shape of CorelDRAW:

```
Sub App()
With ActiveShape.Curve.Subpaths(1)
    MsgBox .EndNode.PositionX
End With
End Sub
```

{button ,AL(`CLS_SubPath;FNC_EndNode')} **Related Topics**

SubPath.FindSegmentOffset

Function **FindSegmentOffset**(ByVal **AbsoluteOffset** AS Double, ByRef **Segment** AS **Segment**, ByRef **ParamOffset** AS Double, ByRef Remainder AS Double) AS Boolean

SubPath

Description

The **FindSegmentOffset** method returns or sets a True or False value that indicates the offset of a segment in COrelDRAW. A segment is a line or curve between nodes in a curve object. There are two types of segments: straight and curved. If you click on a node with the Shape tool, the Status Bar displays the type of segment entering the node and the node type itself.

When you find the segment offset with the **FindSegmentOffset** method, you have the option of passing several values as parameters:

Parameters	Description
AbsoluteOffset	Lets you specify the absolute offset value.
Segment	The **Segment** parameter identifies the segment in the **Segments** collection of CorelDRAW.
ParamOffset	Lets you specify the parametric offset value.
Remainder	Lets you specify the Remainder value. This value is optional.

{button ,AL(`CLS_SubPath;FNC_FindSegmentOffset')} **Related Topics**

Subpaths properties

Subpaths Legend

- Application
- Count
-
- Item
- Parent

Subpaths

Class **Subpaths**

Properties Referenced by

The **Subpaths** class defines the characteristics of **Subpaths** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath. For example, when you convert text to curves, you often create subpaths. Once converted to a curve, the letter "O" appears as two ellipses. Each of these ellipses is a subpath.

You can turn a closed curve object into an open one by breaking its path at any point. You can also break an open path into one or more subpaths or into separate objects.

When you break a path, any subpaths and nodes that are created remain a part of the original object.

{button ,AL(`CLS_Subpaths')} **Related Topics**

Subpaths.Application

Property **Application** AS **Application**

Subpaths

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code examples displays the current version of CorelDRAW in a message box:

```
Sub SubpathApp()
With ActiveShape.Curve.Subpaths
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Subpaths;FNC_Application')} **Related Topics**

Subpaths.Parent

Property **Parent** AS **Shape**

Subpaths

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the length of the subpath collection's parent curve object in a message box. The length of the curve object is measured in document units:

```
Sub SubpathParent()
With ActiveShape.Curve.Subpaths
    MsgBox .Parent.Curve.Length
End With
End Sub
```

{button ,AL(`CLS_Subpaths;FNC_Parent')} **Related Topics**

Subpaths.Item

Property **Item**(ByVal **Index** AS Long) AS **SubPath**

Subpaths

Description

The **Item** property returns a value associated with the index number of a Subpath object in the **Subpaths** collection of CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

`Subpaths(2)` refers to the second subpath in the **Subpaths** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Subpaths.Item(2)` is the same as `Subpaths(2)` - they both reference the second subpath in the **Subpaths** collection.

You must reference a subpath in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Subpaths** collection; it uniquely identifies each member of the collection.

Example

The following code example counts the number of nodes in the first subpath of a selected curve object. A message box displays the number of nodes in a message box:

```
Sub SubpathsItem()
With ActiveShape.Curve.Subpaths
    MsgBox .Item(1).Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_Subpaths;FNC_Item')} **Related Topics**

A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

Subpaths.Count

Property **Count** AS Long

Subpaths

Description

The **Count** property returns the number of subpaths in the **Subpaths** collection of CorelDRAW. A subpath is a curve or shape within a single curve object. A single curve object can consist of more than one curve or shape. Each of these curves or shapes is referred to as a subpath.

Example

The following code example displays the number of subpath objects in the current curve selection in a message box:

```
Sub SubpathCount()
With ActiveShape.Curve.Subpaths
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Subpaths;FNC_Count')} **Related Topics**

Properties properties

Properties Legend

- Count
- Index
- Item
- ItemByIndex

Properties methods

Properties Legend

- Delete
- DeleteByIndex
- Description
- GetFile
- PutFile

Properties

Class **Properties**

Properties Methods Referenced by

The **Properties** class defines the characteristics of object properties in the **Properties** collection of CorelDRAW, and describes and describes the look and behavior of the objects through its properties and methods.

A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

{button ,AL(`CLS_Properties')} **Related Topics**

Properties.Item

Property **Item**(ByVal **Name** AS String, ByVal **ID** AS Long) AS Variant

Properties

Description

The **Item** property returns or sets a value associated with the index number of a property object in the **Properties** collection of CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

The **Item** property of the **Properties** collection is the default property and may be omitted when referencing items in the collection.

For example, `Properties.Item(5)` is the same as `Properties(5)` - they both reference the fifth property in the collection.

Parameters	Description
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.

{button ,AL(`CLS_Properties;FNC_Item')} **Related Topics**

Properties.Index

Property **Index**(ByVal **Name** AS String, ByVal **ID** AS Long) AS Long

Properties

Description

The **Index** property returns a value associated with a property object in the **Properties** collection of CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

`Properties(5)` refers to the fifth property in the **Properties** collection of CorelDRAW.

The **Index** property returns a Read-Only value.

Parameters	Description
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.

{button ,AL(`CLS_Properties;FNC_Index')} **Related Topics**

Properties.ItemByIndex

Property **ItemByIndex**(ByVal **Index** AS Long) AS Variant

Properties

Description

The **ItemByIndex** property returns a value associated with a property, by referencing the index number of the property in CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

This property returns a Read-Only value.

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Properties** collection; it uniquely identifies each member of the collection.

{button ,AL(`CLS_Properties;FNC_ItemByIndex')} **Related Topics**

Properties.Delete

Sub **Delete**(ByVal **Name** AS String, ByVal **ID** AS Long)

Properties

Description

The **Delete** method deletes a property by referencing the name and id number of the property in CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

Parameters	Description
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.

{button ,AL(`CLS_Properties;FNC_Delete')} **Related Topics**

Properties.DeleteByIndex

Sub **DeleteByIndex**(ByVal **Index** AS Long)

Properties

Description

The **DeleteByIndex** method deletes a property by referencing index number of the property within the Properties collection of CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Properties** collection; it uniquely identifies each member of the collection.

{button ,AL(`CLS_Properties;FNC_DeleteByIndex')} **Related Topics**

Properties.Count

Property **Count** AS Long

Properties

Description

The **Count** property returns the number of properties in the **Properties** collection of CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

The **Count** property returns a Read-Only value.

{button ,AL(`CLS_Properties;FNC_Count')} **Related Topics**

Properties.Description

Sub **Description**(ByVal **Index** AS Long, ByRef **Name** AS String, ByRef **ID** AS Long)

Properties

Description

The **Description** method creates a description of a property by specifying the **Index**, **Name** and **ID** of the property in CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Properties** collection; it uniquely identifies each member of the collection.
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.

{button ,AL(`CLS_Properties;FNC_Description')} **Related Topics**

Properties.PutFile

Sub **PutFile**(ByVal **Name** AS String, ByVal **ID** AS Long, ByVal **FileName** AS String)

Properties

Description

The **PutFile** method creates a property and associates it with a file in CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

A file is a document that can be used in CorelDRAW.

Parameters	Description
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.
FileName	The **FileName** parameter identifies the full path name of a document that is used with CorelDRAW.

{button ,AL(`CLS_Properties;FNC_PutFile')} **Related Topics**

Properties.GetFile

Sub **GetFile**(ByVal **Name** AS String, ByVal **ID** AS Long, ByVal **FileName** AS String)

Properties

Description

The **GetFile** method returns a property associated with a file in CorelDRAW. A property is a characteristic attribute, such as size, position, or shape, that defines or describes an object in CorelDRAW.

A file is a document that can be used in CorelDRAW.

Parameters	Description
Name	The **Name** parameter is a string value that uniquely identifies a property in CorelDRAW.
ID	The **ID** parameter is a unique identifier of a property within CorelDRAW.
FileName	The **FileName** parameter identifies the full path name of a document that is used with CorelDRAW.

{button ,AL(`CLS_Properties;FNC_GetFile')} **Related Topics**

Shapes properties

Shapes Legend

- Application

- Count

-
- Item

- Parent

Shapes methods

Shapes Legend

- All
- AllExcluding
- Range

Shapes

Class **Shapes**

Properties Methods Referenced by

The **Shapes** class defines the characteristics of **Shapes** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A shape is any object that can be displayed as several variations of a rectangle or a circle.

Once you have created a shape or line, you can move and shape the object to create the effect you want. Some types of objects, such as rectangles and ellipses, can only be shaped in specific ways. For example, you can round one or more corners of a square or create a pie-shape or arc from a circle.

CorelDRAW provides drawing tools for drawing basic shapes, such as rectangles, ellipses, polygons, stars, grids, and spirals. To draw a shape with one of these tools, drag diagonally in any direction until the shape is the size you want. For each tool, the Status Bar displays the dimensions of the shape as you draw it.

The Shape tool lets you change the shape of all curve objects by editing their nodes and segments. You can also select and edit curves using the Pick tool.

{button ,AL(`CLS_Shapes')} **Related Topics**

Shapes.Application

Property **Application** AS **Application**

Shapes

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ShapesApplication()
With ActiveLayer.Shapes
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_Application')} **Related Topics**

Shapes.Parent

Property **Parent** AS Object

Shapes

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the shape collection's parent object in a message box:

```
Sub ShapesParent()
With ActiveLayer.Shapes
    MsgBox "The name of the shape collection's parent is " & .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_Parent')} **Related Topics**

Shapes.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **Shape**

Shapes

Description

The **Item** property returns a value associated with the index number of a Shape object in the **Shapes** collection of CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

`Shapes(2)` refers to the second shape in the **Shapes** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Shapes.Item(2)` is the same as `Shapes(2)` - they both reference the second shape in the **Shapes** collection.

You must reference a shape in the collection by passing its index number or name as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **Shapes** collection; it uniquely identifies each member of the collection. Name is a string value that uniquely refers to and identifies each member of the **Shapes** collection.

Example

The following code example removes any fill from the first shape object in the **Shapes** collection:

```
Sub ShapesItem()
With ActiveLayer.Shapes
    .Item(1).Fill.ApplyNoFill
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_Item')} **Related Topics**

Shapes.Count

Property **Count** AS Long

Shapes

Description

The **Count** property returns the number of shapes in the **Shapes** collection of CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of shape objects in the **Shapes** collection in a message box:

```
Sub ShapesCount()
With ActiveLayer.Shapes
    MsgBox "There are " & .Count & " shapes in the Shapes collection."
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_Count')} **Related Topics**

Shapes.Range

Function **Range**(ByVal **IndexArray** AS Variant) AS **ShapeRange**

Shapes

Description

The **Range** method returns or sets a value associated with a series of shape objects in the **Shapes** collection in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The **Range** method allows you to reference several shapes concurrently. The use of an index range makes it easier and faster to find, sort, and edit multiple shapes in the **Shapes** collection.

You must reference the shapes in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify specific shapes in the **Shapes** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific shape objects that become members of the array.

Example

The following code example deletes a range of shape objects from the active document. The first (1) shape object and the fourth (4) shape object are deleted from the **Shapes** collection. A message box displays the number of shapes in the collection before and after the deletion, so as to confirm that the shapes were removed from the **Shapes** collection:

```
Sub ShapesRange()
With ActiveLayer.Shapes
    MsgBox "There are " & .Count & " shapes in the Shapes collection."
    .Range(Array(1, 4)).Delete
    MsgBox "There are " & .Count & " shapes in the Shapes collection."
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_Range')} **Related Topics**

An array is a series of objects all of which are the same size and type. Each object in an array is called an array element. For example, you could have an array of integers or an array of characters or an array of anything that has a defined data type. The important characteristics of an array are:

- Each element has the same data type (although they may have different values).
- The entire array is stored contiguously in memory (that is, there are no gaps between elements).
- Arrays can have more than one dimension. A one-dimensional array is called a vector; a two-dimensional array is called a matrix.

Shapes.All

Function **All**() AS **ShapeRange**

Shapes

Description

The **All** method returns or sets a value associated with all shape objects in the **Shapes** collection in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The **All** method allows you to reference all shapes concurrently. The use of the **All** method makes it easier and faster to find, sort, and edit all shapes in the **Shapes** collection.

Example

The following code example selects all shapes in the **Shapes** collection with the **All** method, and applies a two pattern fill to the shapes:

```
Sub ShapesAll()
With ActiveLayer.Shapes
    .All.ApplyPatternFill cdrTwoColorPattern
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_All')} **Related Topics**

Shapes.AllExcluding

Function **AllExcluding**(ByVal **IndexArray** AS Variant) AS **ShapeRange**

Shapes

Description

The **AllExcluding** method returns or sets a value associated with a series of shape objects in the **Shapes** collection in CorelDRAW, with the exception of the shapes identified by their index numbers in the method parameters. A shape is an object that can be displayed as several variations of a rectangle or a circle.

You must reference the shapes in the range by passing an index array as a parameter:

Parameters	Description
IndexArray	An **IndexArray** is a range of index numbers that identify and group specific shapes in the **Shapes** collection. By selecting a range of index numbers in the **Range** method, you are selecting specific shape objects that become members of the array.

Example

The following code example deletes all of the shapes in the **Shapes** collection except the first (1) shape and the second (2) shape, which are excluded from deletion by being passed as an array parameter in the AllExcluding method:

```
Sub ShapesExclude()
With ActiveLayer.Shapes
    .AllExcluding(Array(1, 2)).Delete
    MsgBox "There are " & .Count & " shapes in the Shapes collection."
End With
End Sub
```

{button ,AL(`CLS_Shapes;FNC_AllExcluding')} **Related Topics**

ShapeRange properties

ShapeRange Legend

- Application

- Count

-
- Item

- Parent
  - PositionX
  - PositionY
  - RotationCenterX
  - RotationCenterY
  - SizeHeight
  - SizeWidth

ShapeRange methods

ShapeRange Legend

- Add
- AddRange
- AddToPowerClip
- AddToSelection
- ApplyFountainFill
- ApplyNoFill
- ApplyPatternFill
- ApplyPostscriptFill
- ApplyTextureFill
- ApplyUniformFill
- Clone
- Combine
- ConvertOutlineToObject
- ConvertToBitmap
- ConvertToBitmapEx
- ConvertToCurves
- Copy
- CreateSelection
- Cut
- Delete
- Duplicate
- Flip
- GetBoundingBox
- GetPosition
- GetSize
- Group
- IndexOf
- Move
- OrderBackOf
- OrderBackOne
- OrderForwardOne
- OrderFrontOf
- OrderReverse
- OrderToBack
- OrderToFront
- Remove
- RemoveAll
- RemoveFromContainer
- RemoveFromSelection
- Rotate
- RotateEx
- SetBoundingBox
- SetOutlineProperties
- SetPosition
- SetRotationCenter
- SetSize
- SetSizeEx
- Skew
- SkewEx
- Stretch
- StretchEx

UngroupAll

ShapeRange.GetBoundingBox

Sub **GetBoundingBox**(ByRef **X** As Double, ByRef **Y** As Double, ByRef **Width** As Double, ByRef **Height** As Double, [ByVal UseOutline As Boolean = False])

Gets the shape range bounding box relatively to its lower left corner

Member of **ShapeRange**

Parameters	Description
X	Description of X goes here (**out**)
Y	Description of Y goes here (**out**)
Width	Description of Width goes here (**out**)
Height	Description of Height goes here (**out**)
UseOutline	Description of UseOutline goes here (**in**) Optional Default value = **False**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_GetBoundingBox')} **Related Topics**

ShapeRange.RemoveFromSelection

Sub **RemoveFromSelection**()

Removes the shape in the range from the current selection

Member of **ShapeRange**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_RemoveFromSelection')} **Related Topics**

ShapeRange.SetRotationCenter

Sub **SetRotationCenter**(ByVal **X** As Double, ByVal **Y** As Double)

Sets the rotation center of the shape range

Member of **ShapeRange**

Parameters	Description
X	Description of X goes here (**in**)
Y	Description of Y goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_SetRotationCenter')} **Related Topics**

ShapeRange.SetSizeEx

Sub **SetSizeEx**([ByVal Width As Double = 0], [ByVal Height As Double = 0], ByVal **CenterX** As Double, ByVal **CenterY** As Double)

Sets the shape range size using the anchor point

Member of **ShapeRange**

Parameters	Description
Width	Description of Width goes here (**in**) Optional Default value = **0**
Height	Description of Height goes here (**in**) Optional Default value = **0**
CenterX	Description of CenterX goes here (**in**)
CenterY	Description of CenterY goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_SetSizeEx')} **Related Topics**

ShapeRange.Stretch

Sub **Stretch**(ByVal **StretchX** As Double, [ByVal StretchY As Double = 0], [ByVal StretchCharactersSize As Boolean = False])

Stretches the shapes in the shape range

Member of **ShapeRange**

Parameters	Description
StretchX	Description of StretchX goes here (**in**)
StretchY	Description of StretchY goes here (**in**) Optional Default value = **0**
StretchCharactersSize	Description of StretchCharactersSize goes here (**in**) Optional Default value = **False**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_Stretch')} **Related Topics**

ShapeRange.StretchEx

Sub **StretchEx**(ByVal **StretchX** As Double, [ByVal StretchY As Double = 0], ByVal **CenterX** As Double, ByVal **CenterY** As Double, [ByVal StretchCharactersSize As Boolean = False])

Stretches the shapes in the shape range using the anchor point

Member of **ShapeRange**

Parameters	Description
StretchX	Description of StretchX goes here (**in**)
StretchY	Description of StretchY goes here (**in**) Optional Default value = **0**
CenterX	Description of CenterX goes here (**in**)
CenterY	Description of CenterY goes here (**in**)
StretchCharactersSize	Description of StretchCharactersSize goes here (**in**) Optional Default value = **False**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_ShapeRange;FNC_StretchEx')} **Related Topics**

ShapeRange

Class **ShapeRange**

Properties Methods Referenced by

The **ShapeRange** class defines the characteristics of a range of shape objects and describes the look and behavior of the object range through its properties and methods.

A shape is any object that can be displayed as several variations of a rectangle or a circle.

Once you have opened and formatted your drawing page, you are ready to start drawing. To create your drawing, you will need to know how to create geometrical shapes, straight lines, curves, and irregular shapes.

Once you have created a shape or line, you can move and shape the object to create the effect you want. Some types of objects, such as rectangles and ellipses, can only be shaped in specific ways. For example, you can round one or more corners of a square or create a pie-shape or arc from a circle.

If you want to make more extensive changes to a rectangle, ellipse, or polygon, you can convert the object to a curve. For example, you can create a trapezoid from a rectangle by converting the rectangle to a curve object and then dragging the corners using the Shape tool.

CorelDRAW provides drawing tools for drawing basic shapes, such as rectangles, ellipses, polygons, stars, grids, and spirals. To draw a shape with one of these tools, drag diagonally in any direction until the shape is the size you want. For each tool, the Status Bar displays the dimensions of the shape as you draw it.

Once you have created basic objects, you can use a number of tools, buttons, and commands to change the shape of the object. For example, you can use the Shape tool to round the corners of a rectangle or shape an ellipse into an arc or a pie-shape.

The Shape tool lets you change the shape of all curve objects by editing their nodes and segments. You can also select and edit curves using the Pick tool.

{button ,AL(`CLS_ShapeRange')} **Related Topics**

ShapeRange.CreateSelection

Sub **CreateSelection**()

Description

The **CreateSelection** method creates a single selection from a range of shapes in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Once the selection is made, the shape range selection acts as a single object in CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_CreateSelection')} **Related Topics**

ShapeRange.RemoveFromContainer

Sub **RemoveFromContainer**([ByVal Level As Long = 0])

Description

The **RemoveFromContainer** method removes the shape range from a power clip object in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

When you PowerClip objects, you put one object inside another object or group of objects. One object becomes the contents, while the other becomes the container. You can create a container from any object you create using CorelDRAW, including shapes, lines, curves, Artistic text, and groups. A contents object can be any object you create using CorelDRAW or import from another program.

Parameters	Description
Level	If you created nested PowerClip objects and want to extract all contents objects in succession, you'll need to use perform this task for each nested level. This value is optional and the default value is 0.

{button ,AL(`CLS_ShapeRange;FNC_RemoveFromContainer')} **Related Topics**

ShapeRange.Application

Property **Application** AS **Application**

ShapeRange

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_ShapeRange;FNC_Application')} **Related Topics**

ShapeRange.Parent

Property **Parent** AS **Application**

ShapeRange

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_ShapeRange;FNC_Parent')} **Related Topics**

ShapeRange.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **Shape**

ShapeRange

Description

The **Item** property returns a value associated with the index number of a Shape Range object in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

`ShapeRange(2)` refers to the second shape object in the range.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the range. For example, `ShapeRange.Item(2)` is the same as `ShapeRange(2)` - they both reference the second shape object in the range.

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a ShapeRange class: it uniquely identifies each member of the node range. **Name** is a string value that uniquely identifies the shape in the range.

Example

Example of usage goes here

{button ,AL(`CLS_ShapeRange;FNC_Item')} **Related Topics**

ShapeRange.Count

Property **Count** AS Long

ShapeRange

Description

The **Count** property returns the number of shapes in the ShapeRange class of CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of selected shapes in the active document of CorelDRAW:

```
Sub Test()
    Dim sr As ShapeRange
    Dim s As String
    Set sr = ActiveDocument.Selection
    If sr.Count = 0 Then
        s = "No shapes selected"
    MsgBox s
End Sub
```

{button ,AL(`CLS_ShapeRange;FNC_Count')} **Related Topics**

ShapeRange.Add

Sub **Add**(ByRef **Shape** AS **Shape**)

ShapeRange

Description

The **Add** method places a specified shape object into an existing shape range in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Shape	Specifies the shape object to be added to the shape range in CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_Add')} **Related Topics**

ShapeRange.Remove

Sub **Remove**(ByVal **Index** AS Long)

ShapeRange

Description

The **Remove** method removes a shape object from the shape range in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Index	**Index** is a preset placeholder for each object in the ShapeRange class; it uniquely identifies each member of the range.

{button ,AL(`CLS_ShapeRange;FNC_Remove')} **Related Topics**

ShapeRange.IndexOf

Function **IndexOf**(ByRef **Shape** AS **Shape**) AS Long

ShapeRange

Description

The **IndexOf** property returns the index number associated with a shape in a shape range in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Shape	Specifies a shape in the shape range of CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_IndexOf')} **Related Topics**

ShapeRange.AddToSelection

Sub **AddToSelection**()

ShapeRange

Description

The **AddToSelection** method adds a shape object to a selection in a shape range in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
ResetSelection	Resets the selection of shapes in CorelDRAW. This value is optional and the default value is True.

{button ,AL(`CLS_ShapeRange;FNC_AddToSelection')} **Related Topics**

ShapeRange.ConvertToCurves

Sub **ConvertToCurves**()

ShapeRange

Description

The **ConvertToCurves** method converts the shape objects in a shape range to curve objects in CorelDRAW. A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the ConvertToCurves method of the Shape class.

{button ,AL(`CLS_ShapeRange;FNC_ConvertToCurves')} **Related Topics**

ShapeRange.ConvertToBitmap

Function **ConvertToBitmap**(ByVal BitDepth AS Long, ByVal Grayscale AS Boolean, ByVal Dithered AS Boolean, ByVal TransparentBG AS Boolean, ByVal Resolution AS Long, ByVal AntiAliasing AS **cdrAntiAliasingType**, ByVal UseColorProfile AS Boolean) AS **Shape**

ShapeRange

Description

The **ConvertToBitmap** method converts the shape objects in a shape range to bitmaps in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Parameters	Description
BitDepth	Lets you specify bit depth. This value is optional and the default value is 24.
Grayscale	Set to TRUE (-1) converts to grayscale. This value is optional and the default value is False.
Dithered	Set to TRUE (-1) enables dithering. This value is optional and the default value is True.
TransparentBG	Set to TRUE (-1) enables transparent background. This value is optional and the default value is True.
Resolution	Lets you specify the resolution. This value is optional and the default value is 72.
AntiAliasing	Lets you specify the anti aliasing type.This value is optional and returns cdrAntiAliasingType. The default value is cdrNormalAntiAliasing.
UseColorProfile	Set to TRUE (-1) use the color profile. This value is optional and the default value is True.

{button ,AL(`CLS_ShapeRange;FNC_ConvertToBitmap')} **Related Topics**

ShapeRange.Copy

Sub **Copy**()

ShapeRange

Description

The **Copy** method creates a copy of the shape objects in a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_Copy')} **Related Topics**

ShapeRange.Cut

Sub **Cut**()

ShapeRange

Description

The **Cut** method cuts the shape objects in a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_Cut')} **Related Topics**

ShapeRange.Delete

Sub **Delete**()

ShapeRange

Description

The **Delete** method deletes the shape objects in a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_Delete')} **Related Topics**

ShapeRange.GetPosition

Sub **GetPosition**(ByRef **PositionX** AS Double, ByRef **PositionY** AS Double)

ShapeRange

Description

The **GetPosition** method gets the position of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
PositionX	Sets the horizontal position of the shape range, in document units.
PositionY	Sets the vertical position of the shape range, in document units.

{button ,AL(`CLS_ShapeRange;FNC_GetPosition')} **Related Topics**

ShapeRange.GetSize

Sub **GetSize**(ByRef **Width** AS Double, ByRef **Height** AS Double)

ShapeRange

Description

The **GetSize** method gets the size of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Width	Sets the width of the shape range, in document units.
Height	Sets the height of the shape range, in document units.

{button ,AL(`CLS_ShapeRange;FNC_GetSize')} **Related Topics**

ShapeRange.Move

Sub **Move**(ByVal **DeltaX** AS Double, ByVal **DeltaY** AS Double)

ShapeRange

Description

The **Move** method moves a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
DeltaX	The **DeltaX** parameter specifies the horizontal distance to move a shape range with the **Move** method. Delta refers to a limited incremental value in a variable.
DeltaY	The **DeltaY** parameter specifies the vertical distance to move a shape range with the **Move** method. Delta refers to a limited incremental value in a variable.

{button ,AL(`CLS_ShapeRange;FNC_Move')} **Related Topics**

ShapeRange.Duplicate

Function **Duplicate**(ByVal OffsetX AS Double, ByVal OffsetY AS Double) AS **ShapeRange**

ShapeRange

Description

The **Duplicate** method duplicates a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
OffsetX	The **OffsetX** parameter specifies the horizontal distance to offset the duplicated shape range. This value is optional and the default value is 0.
OffsetY	The **OffsetY** parameter specifies the vertical distance to offset the duplicated shape range. This value is optional and the default value is 0.

{button ,AL(`CLS_ShapeRange;FNC_Duplicate')} **Related Topics**

ShapeRange.Clone

Function **Clone**(ByVal OffsetX AS Double, ByVal OffsetY AS Double) AS **ShapeRange**

ShapeRange

Description

The **Clone** method clones a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

Parameters	Description
OffsetX	The **OffsetX** parameter specifies the horizontal distance to offset the cloned shape range. This value is optional and the default value is 0.
OffsetY	The **OffsetY** parameter specifies the vertical distance to offset the cloned shape range. This value is optional and the default value is 0.

{button ,AL(`CLS_ShapeRange;FNC_Clone')} **Related Topics**

ShapeRange.Group

Function **Group**() AS **Shape**

ShapeRange

Description

The **Group** method creates a group with the active shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_Group')} **Related Topics**

ShapeRange.RemoveAll

Sub **RemoveAll**()

ShapeRange

Description

The **RemoveAll** method removes all shape objects from a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_RemoveAll')} **Related Topics**

ShapeRange.OrderToFront

Sub **OrderToFront**()

ShapeRange

Description

The **OrderToFront** method arranges the stacking order of a shape range by moving it to the front of the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_ShapeRange;FNC_OrderToFront')} **Related Topics**

ShapeRange.OrderToBack

Sub **OrderToBack**()

ShapeRange

Description

The **OrderToBack** method arranges the stacking order of a shape range by moving it to the back of the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_ShapeRange;FNC_OrderToBack')} **Related Topics**

ShapeRange.OrderForwardOne

Sub **OrderForwardOne**()

ShapeRange

Description

The **OrderForwardOne** method arranges the stacking order of a shape range by moving it forward one in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_ShapeRange;FNC_OrderForwardOne')} **Related Topics**

ShapeRange.OrderBackOne

Sub **OrderBackOne**()

ShapeRange

Description

The **OrderBackOne** method arranges the stacking order of a shape range by moving it back one in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_ShapeRange;FNC_OrderBackOne')} **Related Topics**

ShapeRange.OrderFrontOf

Sub **OrderFrontOf**(ByRef **Shape** AS **Shape**)

ShapeRange

Description

The **OrderFrontOf** method arranges the stacking order of a shape range by moving it in front of a specified shape in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

Parameters	Description
Shape	The **Shape** parameter identifies the Shape object that will have another shape placed in front of it in the stacking order of CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_OrderFrontOf')} **Related Topics**

ShapeRange.OrderBackOf

Sub **OrderBackOf**(ByRef **Shape** AS **Shape**)

ShapeRange

Description

The **OrderBackOf** method arranges the stacking order of a shape range by moving it in back of a specified shape in the stacking order in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

Parameters	Description
Shape	The **Shape** parameter identifies the Shape object that will have another shape placed in front of it in the stacking order of CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_OrderBackOf')} **Related Topics**

ShapeRange.OrderReverse

Sub **OrderReverse**()

ShapeRange

Description

The **OrderReverse** method reverses the stacking order of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Stacking order is the sequence in which objects are created in the Image Window. This order determines the relationship between objects and, therefore, the appearance of your image. The first object you create appears on the bottom; the last object appears on the top. You can use the Order commands to place the objects where you want them; however, the background object always appears on the bottom and cannot be reordered.

{button ,AL(`CLS_ShapeRange;FNC_OrderReverse')} **Related Topics**

ShapeRange.Rotate

Sub **Rotate**(ByVal **Angle** AS Double)

ShapeRange

Description

The **Rotate** method rotates a shape range a specified degree in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Angle	The **Angle** parameter specifies, in degrees, the amount to rotate a shape range.

{button ,AL(`CLS_ShapeRange;FNC_Rotate')} **Related Topics**

ShapeRange.RotateEx

Sub **RotateEx**(ByVal **Angle** AS Double, ByVal **CenterX** AS Double, ByVal **CenterY** AS Double)

ShapeRange

Description

The **RotateEx** method rotates a shape range a specified degree in CorelDRAW, by adding a degree of rotation to the original rotation angle.

A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Angle	Sets the angle measurement, in degrees, in which the shape range is rotated.
CenterX	Sets the x coordinate for the center of rotation in the shape range, around which the object rotates.
CenterY	Sets the y coordinate for the center of rotation in the shape range, around which the object rotates.

{button ,AL(`CLS_ShapeRange;FNC_RotateEx')} **Related Topics**

ShapeRange.Skew

Sub **Skew**(ByVal **AngleX** AS Double, ByVal **AngleY** AS Double)

ShapeRange

Description

The **Skew** method skews a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

Parameters	Description
AngleX	The **AngleX** parameter specifies the degree in which to slant the shape range horizontally.
AngleY	The **AngleY** parameter specifies the degree in which to slant the shape range vertically.

{button ,AL(`CLS_ShapeRange;FNC_Skew')} **Related Topics**

ShapeRange.SkewEx

Sub **SkewEx**(ByVal **AngleX** AS Double, ByVal **AngleY** AS Double, ByVal **CenterX** AS Double, ByVal **CenterY** AS Double)

ShapeRange

Description

The **SkewEx** method skews a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Skewing refers to a slant of an object. Skewing a node range slants the range of nodes within a shape object in CorelDRAW.

Parameters	Description
AngleX	Sets the horizontal angle, in degrees, in which the shape range is skewed.
AngleY	Sets the vertical angle, in degrees, in which the shape range is skewed.
CenterX	Sets the x coordinate for the center of rotation in the shape range, around which the object is skewed.
CenterY	Sets the y coordinate for the center of rotation in the shape range, around which the object is skewed.

{button ,AL(`CLS_ShapeRange;FNC_SkewEx')} **Related Topics**

ShapeRange.UngroupAll

Sub **UngroupAll**()

ShapeRange

Description

The **UngroupAll** method removes the grouping feature from all nested groups of the shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_UngroupAll')} **Related Topics**

ShapeRange.Flip

Sub **Flip**(ByVal **Axes** AS **cdrFlipAxes**)

ShapeRange

Description

The **Flip** method flips a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A shape can be flipped horizontally, vertically, or both.

Parameters	Description
Axes	Sets the manner of the flip in the shape object. This value returns cdrFlipAxes.

{button ,AL(`CLS_ShapeRange;FNC_Flip')} **Related Topics**

ShapeRange.PositionX

Property **PositionX** AS Double

ShapeRange

Description

The **PositionX** property returns or sets the horizontal coordinate of a shape range on a page, according to the document's reference point in CorelDRAW.

A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_PositionX')} **Related Topics**

ShapeRange.PositionY

Property **PositionY** AS Double

ShapeRange

Description

The **PositionY** property returns or sets the vertical coordinate of a shape range on a page, according to the document's reference point in CorelDRAW.

A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_PositionY')} **Related Topics**

ShapeRange.SizeWidth

Property **SizeWidth** AS Double

ShapeRange

Description

The **SizeWidth** property returns or set the width of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_SizeWidth')} **Related Topics**

ShapeRange.SizeHeight

Property **SizeHeight** AS Double

ShapeRange

Description

The **SizeHeight** property returns or set the height of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_ShapeRange;FNC_SizeHeight')} **Related Topics**

ShapeRange.RotationCenterX

Property **RotationCenterX** AS Double

ShapeRange

Description

The **RotationCenterX** returns or sets the horizontal coordinate for the center of rotation of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The center of rotation is the point around which an object rotates.

{button ,AL(`CLS_ShapeRange;FNC_RotationCenterX')} **Related Topics**

ShapeRange.RotationCenterY

Property **RotationCenterY** AS Double

ShapeRange

Description

The **RotationCenterY** returns or sets the vertical coordinate for the center of rotation of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The center of rotation is the point around which an object rotates.

{button ,AL(`CLS_ShapeRange;FNC_RotationCenterY')} **Related Topics**

ShapeRange.AddToPowerClip

Sub **AddToPowerClip**(ByRef **Shape** AS **Shape**)

ShapeRange

Description

The **AddToPowerClip** method adds a shape range to a powerclip object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

Parameters	Description
Shape	The **Shape** parameter identifies the shape object that is added to the powerclip in CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_AddToPowerClip')} **Related Topics**

ShapeRange.AddRange

Sub **AddRange**(ByRef **ShapeRange** AS **ShapeRange**)

ShapeRange

Description

The **AddRange** method adds a shape range to an existing shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

In order the use the **AddRange** method, you must pass the **ShapeRange** value as a parameter:

Parameters	Description
ShapeRange	The **ShapeRange** parameter specifies the range of shape items that are added to current shape range.

{button ,AL(`CLS_ShapeRange;FNC_AddRange')} **Related Topics**

ShapeRange.SetPosition

Sub **SetPosition**(ByVal **PositionX** AS Double, ByVal **PositionY** AS Double)

ShapeRange

Description

The **SetPosition** method sets the position of a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
PositionX	Sets the horizontal position of the shape range, in document units.
PositionY	Sets the vertical position of the shape range, in document units.

{button ,AL(`CLS_ShapeRange;FNC_SetPosition')} **Related Topics**

ShapeRange.SetSize

Sub **SetSize**(ByVal Width AS Double, ByVal Height AS Double)

ShapeRange

Description

The **SetSize** method sets the size of a shape range on a page in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

Parameters	Description
Width	Sets the width of the shape range, in document units.
Height	Sets the height of the shape range, in document units.

{button ,AL(`CLS_ShapeRange;FNC_SetSize')} **Related Topics**

ShapeRange.ConvertToBitmapEx

Function **ConvertToBitmapEx**(ByVal Mode AS **cdrImageType**, ByVal Dithered AS Boolean, ByVal Transparent AS Boolean, ByVal Resolution AS Long, ByVal AntiAliasing AS **cdrAntiAliasingType**, ByVal UseColorProfile AS Boolean) AS **Shape**

ShapeRange

Description

The **ConvertToBitmapEx** method converts the shape objects in a shape range to bitmaps in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

A bitmap is an image composed of grids of pixels or dots. Scanners and paint applications such as Corel PHOTO-PAINT generate bitmap images. CorelDRAW creates images using vector objects.

Bitmaps have a fixed resolution so a bitmap looks best when you display or print it at its original size. Enlarging the bitmap appears to enlarge each pixel because extra pixels are added, making the graphic look jagged and distorted. Reducing the size of the bitmap eliminates pixels and shrinks the bitmap.

Parameters	Description
Mode	Sets the image type, and returns cdrImageType. This value is optional and the default value is **cdrRGBColorImage** (4).
Dithered	Lets you specify if the image is dithered when exported. A dithered image uses color that is simulated by putting dots of another color very close together. Windows uses dithering to display colors that the graphics adapter can't display. This value is optional and the default value is False.
Transparent	Sets the transparency with a True or False value. This value is optional and the default value is False.
Resolution	Lets you specify the resolution. This value is optional and the default value is 72.
AntiAliasing	Lets you specify the anti aliasing type.This value is optional and returns cdrAntiAliasingType. The default value is **cdrNormalAntiAliasing**.
UseColorProfile	Set to TRUE (-1) use the color profile. This value is optional and the default value is True.

{button ,AL(`CLS_ShapeRange;FNC_ConvertToBitmapEx')} **Related Topics**

ShapeRange.Combine

Function **Combine**() AS **Shape**

ShapeRange

Description

The **Combine** method combines shape ranges in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

When you combine rectangles, ellipses, polygons, stars, spirals, graphs, or text, CorelDRAW converts them to curves before converting them to a single curve object. When text is combined with other text, however, the text objects are not converted to curves; they are converted into larger blocks of text.

You can break apart any object that has been combined. If you break apart an object that has been created by combining Artistic text, the text breaks apart into lines first, then into words. Paragraph text, on the other hand, breaks into separate paragraphs. Both Artistic and Paragraph text can be recombined.

{button ,AL(`CLS_ShapeRange;FNC_Combine')} **Related Topics**

ShapeRange.SetBoundingBox

Sub **SetBoundingBox**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal **Width** AS Double, ByVal **Height** AS Double, ByVal KeepAspect AS Boolean, ByVal ReferencePoint AS **cdrReferencePoint**)

ShapeRange

Description

The **SetBoundingBox** method sets the bounding box around a shape range in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

The bounding box is the area that surrounds a shape range in CorelDRAW. The bounding box represents the maximum space a shape object can occupy in CorelDRAW. You can resize the bounding box.

Parameters	Description
X	Sets the horizontal position, x coordinate, of the bounding box's center, in document units.
Y	Sets the vertical position, y coordinate, of the bounding box's center, in document units.
Width	Sets the width of the bounding box, in document units.
Height	Sets the height of the bounding box, in document units.
KeepAspect	Keeps the size proportions of the bounding box in relation to its contents. The value is optional and the default value is False.
ReferencePoint	Sets the reference point in CorelDRAW. This value is optional and the default value is cdrCenter (9).

{button ,AL(`CLS_ShapeRange;FNC_SetBoundingBox')} **Related Topics**

ShapeRange.ApplyNoFill

Sub **ApplyNoFill**()

ShapeRange

Description

The **ApplyNoFill** method ensures that no fill is added to a shape range in CorelDRAW. If a fill is currently applied to an object, the **ApplyNoFill** method removes the fill.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

{button ,AL(`CLS_ShapeRange;FNC_ApplyNoFill')} **Related Topics**

ShapeRange.ApplyUniformFill

Sub **ApplyUniformFill**(ByRef **Color** AS **Color**)

ShapeRange

Description

The **ApplyUniformFill** method applies a uniform fill color to a shape range in CorelDRAW. A Uniform colors palette is an independent palette (not based on a color-matching system or your image) that provides 256 colors that are uniformly spread between red, green, and blue.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Parameters	Description
Color	The **Color** parameter defines an existing or new color that is applied as a uniform color to a selected shape object. It uses an existing color component in a palette or a **CreateColor** method (i.e. CreateCMYKColor, CreateRGBColor) to specify a color. A color in a palette is identified by referencing the palette and the index of the color in the palettet. For example, `ActivePalette.Colors(50)`, references the fiftieth color in the active palette of CorelDRAW.

{button ,AL(`CLS_ShapeRange;FNC_ApplyUniformFill')} **Related Topics**

ShapeRange.ApplyFountainFill

Sub **ApplyFountainFill**([ByVal StartColor As **Color** = 0], [ByVal EndColor As **Color** = 0], [ByVal Type As **cdrFountainFillType** = cdrLinearFountainFill (1)], [ByVal Angle As Double = 0], [ByVal Steps As Long = 0], [ByVal EdgePad As Long = 0], [ByVal MidPoint As Long = 50], [ByVal BlendType As **cdrFountainFillBlendType** = cdrDirectFountainFillBlend (0)])

ShapeRange

Description

The **ApplyFountainFill** method applies a fountain fill effect to a shape range in CorelDRAW.A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Parameters	Description
StartColor	The **StartColor** parameter defines the color that appears at the beginning of a fountain fill effect. The value is optional and the default value is Nothing (0).
EndColor	The **EndColor** parameter defines the color that appears at the end of a fountain fill effect. The value is optional and the default value is Nothing (0).
Type	The **Type** parameter defines the fountain fill type in the fill effect. A fountain fill can be linear, conical, radial, or square. The Type parameter returns cdrFountainFillType. This value is optional and the default value is cdrLinearFountainFill (1).
Angle	The **Angle** parameter defines the degree of the angle in linear, conical or square fountain fill effects. Changing the angle of gradation affects the slant of the fountain fill. Radial fountain fills progress in a series of concentric circles, so you can't change their angle. Positive values rotate the fill counterclockwise; negative values rotate it clockwise. This value is optional and the default value is 0.
Steps	The **Steps** parameter identifies the number of bands (steps) used to display a fountain fill. When you create a fountain fill, the space required to blend the colors is divided by the number of fountain steps displayed in the Steps box. This value is optional and the default value is 0.
EdgePad	The **EdgePad** parameter defines the length of solid colors at the beginning and end of the fountain fill before the start blending with the next color in the fountain fill. You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors. This value is optional and the default value is 0.
MidPoint	The **MidPoint** is an imaginary line between two colors in a fountain fill. The value of the mid-point represents its position in relation to two fountain fill colors. By changing this value, you can set the point at which two colors in a fountain fill converge. For example, in a two-color fountain fill using the colors black and white, a value of 50 positions the mid-point in the center of the fill so that half of the fill is black and half is white. Increasing the mid-point value to 99 results in a fountain fill dominated by black. Decreasing the mid-point value to 1 results in a fountain fill dominated by white. This value is optional and the default value is 50.
Blend	The **Blend** parameter determines the blend type in the fountain fill effect. The blend type defines how colors will appear with other colors in the fill effect. This value returns cdrFountainFillBlendType. The value is optional and the default value is cdrDirectFountainFillBlend (0).

{button ,AL(`CLS_ShapeRange;FNC_ApplyFountainFill')} **Related Topics**

ShapeRange.ApplyPatternFill

Sub **ApplyPatternFill**(ByVal **Type** AS **cdrPatternFillType**, ByVal FileName AS String, ByVal PatternCanvasIndex AS Long, ByRef FrontColor AS **Color**, ByRef EndColor AS **Color**, ByVal TransformWithShape AS Boolean)

ShapeRange

Description

The **ApplyPatternFill** method applies a pattern fill effect to a shape range in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Parameters	Description
Type	The **Type** parameter identifies the type of pattern fill and returns a value of cdrPatternFillType.
FileName	The **FileName** parameter identifies the full path name of the graphic that is imported into CorelDRAW to be used as a pattern fill. A file name contains the computer's path and the name of the graphic file. This value is optional.
PatternCanvasIndex	The **PatternCanvasIndex** parameter identifies the pattern canvas in a pattern fill effect. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. Each canvas has an index number that uniquely identifies the canvas. This value is optional.
FrontColor	The **FrontColor** parameter identifies the foreground color in a pattern fill. This value is optional and the default value is Nothing(0).
EndColor	The **EndColor** parameter identifies the background color in a pattern fill. This value is optional and the default value is Nothing(0).
TransformWithShape	The **TransformWithShape** parameter is a True or False value, indicating if the pattern fill changes to fit its shape when the shape object is altered. If the value is True, the pattern fill will change according to changes in its shape object. This value is optional and the default value is False.

{button ,AL(`CLS_ShapeRange;FNC_ApplyPatternFill')} **Related Topics**

ShapeRange.ApplyTextureFill

Sub **ApplyTextureFill**(ByVal **TextureName** AS String, ByVal LibraryName AS String)

ShapeRange

Description

The **ApplyTextureFill** method applies a texture fill effect to a shape range in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Parameters	Description
TextureName	The **TextureName** property returns a string value associated with the name of a texture fill pattern in CorelDRAW. For example, "Air Brush" is texture fill in the "Samples" library of CorelDRAW.
LibraryName	The **LibraryName** parameter returns a string value associated with the name of a texture fill pattern library in CorelDRAW. A texture library is a collection of texture fill pattern files in CorelDRAW. There are several texture libraries included in CorelDRAW. For example, "Samples" is a texture library. The LibraryName parameter is optional.

{button ,AL(`CLS_ShapeRange;FNC_ApplyTextureFill')} **Related Topics**

ShapeRange.ApplyPostscriptFill

Sub **ApplyPostscriptFill**(ByVal **IndexOrName** AS Variant)

ShapeRange

Description

The **ApplyPostscriptFill** method applies a postscript fill effect to a shape range in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language. Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Parameters	Description
IndexOrName	The **Index** parameter specifies the index number of a postscript fill object in CorelDRAW. The **Index** parameter uniquely identifies a postscript fill. The **Name** parameter is the string value that uniquely identifies the postscript fill.

{button ,AL(`CLS_ShapeRange;FNC_ApplyPostscriptFill')} **Related Topics**

ShapeRange.ConvertOutlineToObject

Sub **ConvertOutlineToObject**()

ShapeRange

Description

The **ConvertOutlineToObject** method creates a new shape object from an outline of a shape object in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle.

An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill:

{button ,AL(`CLS_ShapeRange;FNC_ConvertOutlineToObject')} **Related Topics**

ShapeRange.SetOutlineProperties

Sub **SetOutlineProperties**(ByVal **Width** AS Double, ByRef Style AS **OutlineStyle**, ByRef Color AS **Color**, ByRef StartArrow AS **ArrowHead**, ByRef EndArrow AS **ArrowHead**, ByVal BehindFill AS Boolean, ByVal ScaleWithShape AS Boolean, ByVal LineCaps AS **cdrOutlineLineCaps**, ByVal LineJoin AS **cdrOutlineLineJoin**, ByVal NibAngle AS Double, ByVal NibStretch AS Long)

ShapeRange

Description

The **SetOutlineProperties** method allows you to set all the properties of an outline object in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill:

Parameters	Description
Width	The **Width** property returns or sets a numerical value that indicates the width of an object's outline in CorelDRAW. An outline's width is measured in points.
Style	The **Style** property returns or sets the outline style of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
Color	The **Color** property returns or sets the color of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
StartArrow	The **StartArrow** property returns or sets the start arrow of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
EndArrow	The **EndArrow** property returns or sets the end arrow of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
BehindFill	The **BehindFill** property returns a True or False value that indicates if an outline's fill is behind the outline in CorelDRAW. This value is optional and the default value is False.
ScaleWithShape	The **ScaleWithShape** property returns or sets a True or False value that indicates if an object's outline maintains the size proportions of the object in CorelDRAW. This value is optional and the default value is False.
LineCaps	The **LineCaps** property returns or sets the endpoint style of an open path outline in CorelDRAW. This value is optional. It returns a value of cdrOutlineLineCaps and the default value is cdrOutlineButtLineCaps (0)
LineJoin	The **LineJoin** property sets the appearance of lines that intersect each other in an outline. This value returns cdrOutlineLineJoin. The default value is cdrOutlineMiterLineJoin.
NibAngle	The **NibAngle** property returns or sets a numerical value that contributes to the angle of an object's outline in CorelDRAW. This value is optional and the default value is 0.
NibStretch	The **NibStretch** property returns or sets a numerical value that contributes to the thickness of an object's outline in CorelDRAW. This value is optional and the default value is 100.

{button ,AL(`CLS_ShapeRange;FNC_SetOutlineProperties')} **Related Topics**

Layers properties

Layers Legend

- Application
- Count
-
- Item
- Parent

Layers

Class **Layers**

Properties Referenced by

The **Layers** class defines the characteristics of layer collection objects and describes the look and behavior of the objects through its properties and methods.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The layering feature of CorelDRAW gives you added flexibility for organizing and editing the objects in your drawings. You can divide a drawing into multiple layers, each containing a portion of the drawing's contents. Together, layers act as a hierarchy that helps determine the vertical arrangement of a drawing's components. In this arrangement (the stacking order), objects on the top layer always overlay objects on the layer below.

The Layer Manager view lists all the layers in your document, without displaying sublevels or objects. It is the simplest view available, and is an easy way to switch between layers or move objects between layers.

You can assign a new name to any layer you create in the Object Manager.You can edit objects on any unlocked layer. You can also move and copy objects between any layers that are unlocked, as long as they are on the same page (or on the Master Page and another page).If you disable the Edit Across Layers function, you can work only on the active layer and the Desktop layer - you can't select or edit objects on inactive layers.

Each new file, regardless of how many pages it contains, has one master page. This master page controls four default layers: the Grid, Guides, and Desktop layers, plus one Layer (called Layer1) for drawing. To use a layer in the drawing, you must first make the layer active. Once active, a layer is ready to receive any new objects you draw, import, or paste onto it. In the Object Manager, the active layer appears highlighted in red. When you start a drawing, the default layer (Layer 1) is the active layer.

The Delete command removes the current layer highlighted in the Object Manager.

{button ,AL(`CLS_Layers')} **Related Topics**

Layers.Application

Property **Application** AS **Application**

Layers

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub LayersApp()
With ActivePage.Layers
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Layers;FNC_Application')} **Related Topics**

Layers.Parent

Property **Parent** AS **Page**

Layers

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the layer's parent page in a message box:

```
Sub LayersParent()
With ActivePage.Layers
    MsgBox .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Layers;FNC_Parent')} **Related Topics**

Layers.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **Layer**

Layers

Description

The **Item** property returns a value associated with the index number of a layer object in the **Layers** collection of CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain.

`Layers(2)` refers to the second layer object in the **Layers** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Layers.Item(2)` is the same as `Layers(2)` - they both reference the second layer object in the **Layers** collection.

You must reference a layer in the collection by passing its index number as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **Layers** collection; it uniquely identifies each member of the collection. **Name** is the unique string value that identifies each layer.

Example

The following code example changes the name of the first layer in the **Layers** collection and displays the new layer name in a message box:

```
Sub LayersItem()
With ActivePage.Layers
    .Item(1).Name = "CorelDRAW First Layer"
    MsgBox .Item(1).Name
End With
End Sub
```

{button ,AL(`CLS_Layers;FNC_Item')} **Related Topics**

Layers.Count

Property **Count** AS Long

Layers

Description

The **Count** property returns the number of layers in the **Layers** collection of CorelDRAW. A document page is sometimes called a page. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of layers in the **Layers** collection in a message box:

```
Sub LayersCount()
With ActivePage.Layers
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Layers;FNC_Count')} **Related Topics**

Pages properties

Pages Legend

- Application
- Count
-
- Item
- Parent

Pages

Class **Pages**

Properties Referenced by

The **Pages** class defines the characteristics of page collection objects and describes the look and behavior of the objects through its properties and methods.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The default page orientation that you choose - landscape or portrait - automatically applies to every page in a multi-page document. Any new pages you add after creating the document are automatically assigned the same orientation.

However, you can change the orientation of an individual page within a multi-page document without affecting the orientation of other pages using the default.

CorelDRAW provides an array of preset page sizes to choose from, including standard North American and European sizes, and it also lets you create your own custom page sizes. In a multi-page document, the page size you assign anywhere in your document becomes the default size for all pages. Any new pages you add to the document later are automatically assigned the same default page size.

However, you can give individual pages within a multi-page document a custom size, without affecting other pages that are using the default.

{button ,AL(`CLS_Pages')} **Related Topics**

Pages.Application

Property **Application** AS **Application**

Pages

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PagesApp()
With ActiveDocument.Pages
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Pages;FNC_Application')} **Related Topics**

Pages.Parent

Property **Parent** AS **Document**

Pages

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the page's parent layer in a message box:

```
Sub PagesParent()
With ActiveDocument.Pages
    MsgBox .Parent.ActiveLayer.Name
End With
End Sub
```

{button ,AL(`CLS_Pages;FNC_Parent')} **Related Topics**

Pages.Item

Property **Item**(ByVal **Index** AS Long) AS **Page**

Pages

Description

The **Item** property returns a value associated with the index number of a page object in the **Pages** collection of CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page.

`Pages(2)` refers to the second page object in the **Pages** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Pages.Item(2)` is the same as `Pages(2)` - they both reference the second page object in the **Pages** collection.

A page with an index number of 0 returns the Master Page in CorelDRAW.

You must reference a page in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **Pages** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the name of the first page in the **Pages** collection in a message box:

```
Sub PagesItem()
With ActiveDocument.Pages
    MsgBox .Item(1).Name
End With
End Sub
```

{button ,AL(`CLS_Pages;FNC_Item')} **Related Topics**

A page containing information that you want to appear on every page of a multi-page document.

Pages.Count

Property **Count** AS Long

Pages

Description

The **Count** property returns the number of pages in the **Pages** collection of CorelDRAW. A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of pages in the **Pages** collection in a message box:

```
Sub PagesCount()
With ActiveDocument.Pages
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Pages;FNC_Count')} **Related Topics**

Documents properties

Documents Legend

- Application
- Count
-
- Item
- Parent

Documents

Class **Documents**

Properties Referenced by

The **Documents** class defines the characteristics of Document collection objects and describes the look and behavior of the collection objects through its properties and methods. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is often referred to as a page. You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

New documents use the preset options built into the CorelDRAW application. Every new document allows you to save settings so that new drawings opened in CorelDRAW open to the settings established in this Document page. These options include:

- Options set in the Options dialog box
- Toolbar settings
- Docker window
- Color palette

You can add, delete, and rename pages in documents as well as create web documents. Detailed information about each CorelDRAW document can be viewed and saved in a text file within a word processing application for future use.

{button ,AL(`CLS_Documents')} **Related Topics**

Documents.Application

Property **Application** AS **Application**

Documents

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub DocApp()
With Documents
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Documents;FNC_Application')} **Related Topics**

Documents.Parent

Property **Parent** AS **Application**

Documents

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the **Documents** collection's parent layer in a message box:

```
Sub DocParent()
With Documents
    MsgBox .Parent.ActiveLayer.Name
End With
End Sub
```

{button ,AL(`CLS_Documents;FNC_Parent')} **Related Topics**

Documents.Item

Property **Item**(ByVal **Index** AS Long) AS **Document**

Documents

Description

The **Item** property returns a value associated with the index number of a document in the **Documents** collection of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

`Documents(2)` refers to the second effect in the **Documents** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Documents.Item(2)` is the same as `Documents(2)` - they both reference the second document in the **Documents** collection.

You must reference a document in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each document in a **Documents** collection; it uniquely identifies each member of the collection.

Example

The following code example sets a new name for the first document in the **Documents** collection in CorelDRAW. The new document name displays in a message box:

```
Sub DocsItem()
With Documents
    .Item(1).Name = "CorelDRAW_Document"
End With
End Sub
```

{button ,AL(`CLS_Documents;FNC_Item')} **Related Topics**

Documents.Count

Property **Count** AS Long

Documents

Description

The **Count** property returns the number of document objects in the **Documents** collection of CorelDRAW.A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of documents in the **Documents** collection in a message box:

```
Sub DocsCount()
With Documents
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Documents;FNC_Count')} **Related Topics**

Window properties

Window Legend

- Active
- ActiveView
- Application
- Caption
- Document
  - FullScreen
  - Height
- Index
  - Left
- Next
- Page
- Parent
- Previous
  - Top
  - Width
  - WindowState

Window methods

Window Legend

- Activate
- Close
- NewWindow
- Refresh

Window

Class **Window**

Properties Methods Referenced by

The **Window** class defines the characteristics of window objects and describes the look and behavior of the objects through its properties and methods.

A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

You can have several windows open at the same time in the CorelDRAW.

{button ,AL(`CLS_Window')} **Related Topics**

A Window contains a CorelDRAW drawing. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print.

Window.Application

Property **Application** AS Object

Window

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub WindowApplication()
With ActiveWindow.Application
    MsgBox .Version
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Application')} **Related Topics**

Window.Parent

Property **Parent** AS **Windows**

Window

Description

The **Parent** property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the caption of the window's parent object in a message box:

```
Sub WindowParent()
With ActiveWindow
    MsgBox .ActiveView.Parent.Caption
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Parent')} **Related Topics**

Window.Activate

Sub **Activate**()

Window

Description

The **Activate** method opens a window in CorelDRAW and makes it the active window, if it is not currently open. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

In may cases, newer operating systems like Windows 98 and Windows 2000 don't allow the application to activate itself, to prevent intervention with you're current activity. If the application is not granted active status in certain situations, the application window will start flashing in the taskbar, but it will not appear in the foreground.

Example

The following code example activates the current window in CorelDRAW, if it is not already open:

```
Sub WindowActivate()
If ActiveWindow.Active = False Then
    ActiveWindow.Activate
End If
End Sub
```

{button ,AL(`CLS_Window;FNC_Activate')} **Related Topics**

Window.Close

Sub **Close**()

Window

Description

The **Close** method closes the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

Before you close an active file, CorelDRAW prompts you to save the file. You can also close specific windows, as well as close all open files or views:

Example

The following code example closes the active window, if the view type of that window is the Enhanced View:

```
Sub WindowClose()
With ActiveWindow
    If .ActiveView.Type = cdrEnhancedView Then
    .Close
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Close')} **Related Topics**

Window.FullScreen

Property **FullScreen** AS Boolean

Window

Description

The **FullScreen** property returns a True or False value indicating if the active window is visible on a full computer screen. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

If the property is set to True, the active window fills up the full computer screen. If the value is False, the active window is viewed in the normal application window.

Example

The following code example sets the screen view of the drawing window to fit the entire screen, if the view type of the current window is set to Enhanced View:

```
Sub WindowFullScreen()
If ActiveWindow.ActiveView.Type = cdrEnhancedView Then
    ActiveWindow.FullScreen = True
End If
End Sub
```

{button ,AL(`CLS_Window;FNC_FullScreen')} **Related Topics**

Window.Page

Property **Page** AS Object

Window

Description

The **Page** property returns the active page of a document into the active window of CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

A window may contain several document pages, simply called a "page." You can access a document page in a multipage document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Page** property returns a Read-Only value.

Example

The following code example displays the name of the active window's current page in a message box:

```
Sub WindowPage()
With ActiveWindow.Page
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Page')} **Related Topics**

Window.Active

Property **Active** AS Boolean

Window

Description

The **Active** property returns a True or False value indicating the activity status of a window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

If the window is active, a value of True is returned, indicating that it is currently in use. If the window is inactive, the return value is False, indicating that the window is currently open but not in use.

Use the **Activate** method to change the activity status of the current window.

The **Active** property returns a Read-Only value.

Example

The following code example displays the activity status of the current window in a message box:

```
Sub WindowActive()
MsgBox ActiveWindow.Active
End Sub
```

{button ,AL(`CLS_Window;FNC_Active')} **Related Topics**

Window.Caption

Property **Caption** AS String

Window

Description

The **Caption** property returns the name of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The caption appears as text in the title bar of the main application window. If the window is maximized when using the **Caption** property, the document title also displays after the caption in the title bar.

The **Caption** property returns a Read-Only value.

Example

The following code example displays the caption of the active window in a message box:

```
Sub WindowCaption()
With ActiveWindow
    MsgBox .Caption
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Caption')} **Related Topics**

Window.Height

Property **Height** AS Long

Window

Description

The **Height** property returns or sets the height of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The height of the window is a numerical value measured in screen pixels and measures from the top to the bottom of the window.

Example

The following code example sets the zoom of the current view to 200% if the height of the active window is greater than 200 pixels:

```
Sub WindowHeight()
With ActiveWindow
If .Height > 200 Then
ActiveWindow.ActiveView.Zoom = 200
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Height')} **Related Topics**

Window.Width

Property **Width** AS Long

Window

Description

The **Width** property returns or sets the width of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The width of the window is a numerical value measured in screen pixels and measures from the left to the right of the window.

Example

The following code example sets the zoom of the current view to 200% if the width of the active window is greater than 500 pixels:

```
Sub WindowWidth()
With ActiveWindow
If .Width > 500 Then
ActiveWindow.ActiveView.Zoom = 200
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Width')} **Related Topics**

Window.Left

Property **Left** AS Long

Window

Description

The **Left** property sets the location of the left border of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The **Left** property sets the distance, in screen pixels, from the active window's left border to the left side of main application window.

Example

The following code example sets the zoom of the current view to 200% if the distance from the left side of the active window to the left border of the application window is more than 20 pixels:

```
Sub WindowLeft()
With ActiveWindow
If .Left > 20 Then
ActiveWindow.ActiveView.Zoom = 200
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Left')} **Related Topics**

Window.Top

Property **Top** AS Long

Window

Description

The **Top** property sets the location of the left border of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The **Top** property sets the distance, in screen pixels, from the active window's top border to the top of the application window's border in CorelDRAW.

Example

The following code example sets the zoom of the current view to 200% if the distance from the top of the active window to the top border of the application window is more than 100 pixels:

```
Sub WindowTop()
With ActiveWindow
If .Top > 100 Then
ActiveWindow.ActiveView.Zoom = 200
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Top')} **Related Topics**

Window.WindowState

Property **WindowState** AS **cdrWindowState**

Window

Description

The **WindowState** property returns or sets the window state of the current window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

A window can be minimized, maximized or in a normal state.

The **WindowState** property returns a value of cdr WindowState.

Example

The following code example minimizes the active window if the view type is set to Enhanced View:

```
Sub WindowState()
With ActiveWindow
    If .ActiveView.Type = cdrEnhancedView Then
    .WindowState = cdrWindowMinimized
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_WindowState')} **Related Topics**

Window.Previous

Property **Previous** AS **Window**

Window

Description

The **Previous** method returns the window in the **Windows** collection that precedes the selected window object in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

For example, if the selected window object has an index value of 2, the **Previous** method returns the window with an index value of 1 in the **Windows** collection.

The **Previous** property returns a Read-Only value.

Example

The following code example displays the caption of the window that precedes the active window in the **Windows** collection:

```
Sub WindowPrevious()
With ActiveWindow
    MsgBox .Previous.Caption
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Previous')} **Related Topics**

Window.Next

Property **Next** AS **Window**

Window

Description

The **Next** method returns the window in the **Windows** collection that follows the selected window object in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

For example, if the selected window object has an index value of 2, the **Next** method returns the window with an index value of 3 in the **Windows** collection.

The **Next** property returns a Read-Only value.

Example

The following code example displays the caption of the window that follows the active window in the **Windows** collection:

```
Sub WindowPrevious()
With ActiveWindow
    MsgBox .Next.Caption
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Next')} **Related Topics**

Window.Index

Property **Index** AS Long

Window

Description

The **Index** property returns a value associated with a window object in the **Windows** collection of CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

Every window in CorelDRAW is identified by an index number. For example, `Windows(2)`, or Window.Index (2) both refer to the second window in the **Windows** collection of CorelDRAW. The number 2 is the window's index number.

The **Index** property returns a Read-Only value.

Example

The following code example closes the active window if the index of the active window is 1:

```
Sub WindowIndex()
With ActiveWindow
    If .Index = 1 Then
    .Close
End If
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Index')} **Related Topics**

Window.NewWindow

Function **NewWindow**() AS **Window**

Window

Description

The **NewWindow** method creates a new window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

Example

The following code example creates a new window in CorelDRAW. A message displays the number of open windows, including the new window, in a message box:

```
Sub WindowNew()
ActiveWindow.NewWindow
    MsgBox Windows.Count
End Sub
```

{button ,AL(`CLS_Window;FNC_NewWindow')} **Related Topics**

Window.Refresh

Sub **Refresh**()

Window

Definition

The **Refresh** method refreshes the active window in CorelDRAW, updating the window with the most recent information. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

Example

The following code example updates the active window with the **Refresh** method:

```
Sub WindowRefresh()
ActiveWindow.Refresh
End Sub
```

{button ,AL(`CLS_Window;FNC_Refresh')} **Related Topics**

Window.Document

Property **Document** AS **Document**

Window

Definition

The **Document** property returns a value associated with a document contained in the active window of CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

A document is a single page drawing or a collection of single-page drawings that make up a multi-page document.

The **Document** property returns a Read-Only value.

Example

The following code example displays the name of the active window's document in a message box:

```
Sub WindowDocument()
With ActiveWindow.Document
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_Document')} **Related Topics**

Window.ActiveView

Property **ActiveView** AS **ActiveView**

Window

Description

The **ActiveView** property returns a value associated with the view type of the active window in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

Each view type value is associated with a view quality in CorelDRAW.

The **ActiveView** property is a Read-Only value.

Example

The following code example displays the view type of the active window in CorelDRAW:

```
Sub WindowActiveView()
With ActiveWindow.ActiveView
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Window;FNC_ActiveView')} **Related Topics**

Windows properties

Windows Legend

- Application
- Count
-
- Item
- Parent

Windows methods

Windows Legend

- Arrange
- CloseAll
- Refresh

Windows

Class **Windows**

Properties Methods Referenced by

The **Windows** class defines the characteristics of **Windows** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

You can have several windows open at the same time in the CorelDRAW.

{button ,AL(`CLS_Windows')} **Related Topics**

Windows.Application

Property **Application** AS Object

Windows

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ApplicationWindows()
With Windows
MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Windows;FNC_Application')} **Related Topics**

Windows.Parent

Property **Parent** AS Object

Windows

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the path of the window collection's parent object in a message box:

```
Sub ParentWindows()
With Windows
MsgBox .Parent.Path
End With
End Sub
```

{button ,AL(`CLS_Windows;FNC_Parent')} **Related Topics**

Windows.Item

Property **Item**(ByVal **Index** AS Long) AS **Window**

Windows

Description

The **Item** property returns a value associated with the index number of a window object in the **Windows** collection of CorelDRAW.A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

`Windows(2)` refers to the second window in the **Windows** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Windows.Item(2)` is the same as `Windows(2)`. Both structures reference the second window in the **Windows** collection.

You must reference a window in the collection by passing its index number as a parameter:

Parameters	Description
Index	Index is a preset placeholder for each object in a Windows collection; it uniquely identifies each member of the collection.

Example

The following code example displays the caption of the second window in the **Windows** collection in a message box:

```
Sub WindowsItem()
MsgBox Windows.Item(2).Caption
End Sub
```

{button ,AL(`CLS_Windows;FNC_Item')} **Related Topics**

Windows.Count

Property **Count** AS Long

Windows

Description

The **Count** property returns the number of windows in the **Windows** collection of CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of windows in **Windows** collection of CorelDRAW:

```
Sub WindowsCount()
MsgBox Windows.Count
End Sub
```

{button ,AL(`CLS_Windows;FNC_Count')} **Related Topics**

Windows.CloseAll

Sub **CloseAll**()

Windows

Description

The **CloseAll** method closes all open windows in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

You are prompted to save your changes in each window when you execute the **CloseAll** method.

Example

The following code example uses the **CloseAll** method to exit all the open windows in CorelDRAW:

```
Sub WindowsClose()
Windows.CloseAll
End Sub
```

{button ,AL(`CLS_Windows;FNC_CloseAll')} **Related Topics**

Windows.Arrange

Sub **Arrange**(ByVal **Style** AS **cdrWindowArrangeStyle**)

Windows

Description

The **Arrange** method organizes all open windows in CorelDRAW. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

A window can be tiled horizontally or vertically, as well as cascaded in the application window.

You must reference the style in which to organize the windows by passing the style type as a parameter:

Parameters	Description
Style	The **Style** parameter identifies the type of window arrangement in CorelDRAW. The styles for windows in CorelDRAW are Tile Vertically, Tile Horizontally, and Cascade.

Example

The following code example uses the **Arrange** method to cascade all open windows in CorelDRAW:

```
Sub WindowsArrange()
Windows.Arrange (cdrCascade)
End Sub
```

{button ,AL(`CLS_Windows;FNC_Arrange')} **Related Topics**

cdrTileHorizontally=0

cdrTileVertically=1

cdrCascade=2

Windows.Refresh

Sub **Refresh**()

Windows

Definition

The **Refresh** method refreshes all windows in the Windows collection of CorelDRAW, updating the windows with the most recent information. A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

Example

The following code example uses the **Refresh** method to update the active window in CorelDRAW with the most current information:

```
Sub WindowsRefresh()
Windows.Refresh
End Sub
```

{button ,AL(`CLS_Windows;FNC_Refresh')} **Related Topics**

AppWindow properties

AppWindow Legend

- Active

- Application
  - Caption

- ClientHeight

- ClientWidth

- Handle
  - Height
  - Left

- Parent
  - Top
  - Width
  - WindowState

AppWindow methods

AppWindow Legend

- Activate

AppWindow

Class **AppWindow**

Properties Methods Referenced by

The **AppWindow** class defines the characteristics of the application window object describes the look and behavior of the object through its properties and methods. The application window represents the main CorelDRAW window - it acts as a container for the application.

The large white portion of the CorelDRAW application window is the Drawing Window. The rectangle in the center with the drop shadow is the Drawing Page. Usually, only the part of your drawing that falls within the Drawing Page is printed. You can use the remaining space, called the Desktop layer, in the Drawing Window to keep your tools and pieces of your drawing handy.

Common Features in the CorelDRAW application window

- Toolbox
- Property bar
- Workspaces
- Docker window

{button ,AL(`CLS_AppWindow')} **Related Topics**

AppWindow.Handle

Property **Handle** As Long

Description

The **Handle** property returns a value associated with the handle of the window in CorelDRAW. A handle allows you to resize the window in CorelDRAW.

The application window represents the main CorelDRAW window - it acts as a container for the application. This property returns a Read-Only value.

{button ,AL(`CLS_AppWindow;FNC_Handle')} **Related Topics**

Contains a CorelDRAW drawing. You can draw anywhere in the Drawing Window, but only objects that appear on the Drawing Page (indicated by a rectangle with a drop shadow) print.

The portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

AppWindow.Application

Property **Application** AS Object

AppWindow

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub VersionApplication()
With AppWindow
    MsgBox "You are using CorelDRAW " & Version
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Application')} **Related Topics**

AppWindow.Parent

Property **Parent** AS Object

AppWindow

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the path of the CorelDRAW application in a message box:

```
Sub AppWindowParent()
With AppWindow
    MsgBox .Parent.Application.Path
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Parent')} **Related Topics**

AppWindow.Activate

Sub **Activate**()

AppWindow

Description

The **Activate** method opens the main application window in CorelDRAW, if the window is not currently open. The application window represents the main CorelDRAW window - it acts as a container for the application.

In may cases, newer operating systems like Windows 98 and Windows 2000 don't allow the application to activate itself, to prevent intervention with you're current activity. If the application is not granted active status in certain situations, the application window will start flashing in the taskbar, but it will not appear in the foreground.

Example

The following code example activates the CorelDRAW main application window if it is currently inactive. A window is inactive if it is open, but it is not the currently used window application:

```
Sub Activate()
If Not AppWindow.Active Then
    AppWindow.Activate
End If
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Activate')} **Related Topics**

AppWindow.Active

Property **Active** AS Boolean

AppWindow

Description

The **Active** property returns a True or False value indicating the status of the CorelDRAW main application window. If the window is active, a value of true is returned, indicating that it is currently in use. The application window represents the main CorelDRAW window - it acts as a container for the application.

If the window is inactive, the return value is false, indicating that the main application window is currently open but not in use.

Use the **Activate** method to change the inactive status of the main application window.

The **Active** property returns a Read-Only value.

Example

The following code example displays the status of the CorelDRAW main application window in a message box:

```
Sub AppWindowActive()
If AppWindow.Active Then
    MsgBox "CorelDRAW is currently active."
Else
    MsgBox "CorelDRAW is not active at this time."
End If
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Active')} **Related Topics**

AppWindow.Caption

Property **Caption** AS String

AppWindow

Description

The **Caption** property returns or sets the display name of the CorelDRAW main application window. The caption appears as text in the title bar of the application window. The application window represents the main CorelDRAW window - it acts as a container for the application.

If the document window is maximized when using the **Caption** property, the document title will also display after the caption in the title bar of the main application window.

Examples

The following code example sets the caption of the CorelDRAW main application window:

```
Sub AppWindowCaption()
AppWindow.Caption = "My Draw Application"
End Sub
```

The following code example changes the existing caption of the CorelDRAW main application window:

```
Sub ResetCaption()
Dim NewCaption As String
NewCaption = "CorelDRAW"
If AppWindow.Caption <> NewCaption Then
    AppWindow.Caption = NewCaption
End If
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Caption')} **Related Topics**

AppWindow.Height

Property **Height** AS Long

AppWindow

Description

The **Height** property sets the height of the CorelDRAW main application window on your desktop. The application window represents the main CorelDRAW window - it acts as a container for the application.

This property only applies when the main application window is in a normal window state. This property will not work when the application window is minimized or maximized.

The height of the application window is a numerical value measured in screen pixels.

Example

The following code example sets the current window state to normal, defines the new dimensions of the window, resizes the CorelDRAW main application window according to the new dimensions:

```
Sub AppWindowHeight()
With AppWindow
    .WindowState = cdrWindowNormal
    .Top = 10
    .Left = 10
    .Height = 500
    .Width = 600
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Height')} **Related Topics**

cdrWindowNormal=1
cdrWindowMaximized=3
cdrWindowMinimized=2
cdrWindowRestore=9

AppWindow.Width

Property **Width** AS Long

AppWindow

Description

The **Width** property sets the height of the CorelDRAW main application window on your desktop. The application window represents the main CorelDRAW window - it acts as a container for the application.

This property only applies when the main application window is in a normal window state. This property will not work when the application window is minimized or maximized.

The width of the application window is a numerical value measured in screen pixels.

Example

The following code example sets the current window state to normal, defines the new dimensions of the window, resizes the CorelDRAW main application window according to the new dimensions:

```
Sub AppWindowWidth()
With AppWindow
    .WindowState = cdrWindowNormal
    .Top = 10
    .Left = 10
    .Height = 500
    .Width = 600
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Width')} **Related Topics**

AppWindow.Left

Property **Left** AS Long

AppWindow

Description

The **Left** property sets the location of the left border of the CorelDRAW main application window on your desktop. The application window represents the main CorelDRAW window - it acts as a container for the application.

This property only applies when the main application window is in a normal window state. This property will not work when the application window is minimized or maximized.

The **Left** property sets the distance, in screen pixels, from the CorelDRAW window's left border to the left side of the monitor's display area.

Example

The following code example sets the current window state to normal, defines the new dimensions of the window, resizes the CorelDRAW main application window according to the new dimensions:

```
Sub AppWindowLeft()
With AppWindow
    .WindowState = cdrWindowNormal
    .Top = 10
    .Left = 10
    .Height = 500
    .Width = 600
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Left')} **Related Topics**

AppWindow.Top

Property **Top** AS Long

AppWindow

Description

The **Top** property sets the location of the left border of the CorelDRAW main application window on your desktop. The application window represents the main CorelDRAW window - it acts as a container for the application.

This property only applies when the main application window is in a normal window state. This property will not work when the application window is minimized or maximized.

The **Top** property sets the distance, in screen pixels, from the CorelDRAW window's top border to the top of the monitor's display area.

Example

The following code example sets the current window state to normal, defines the new dimensions of the window, resizes the CorelDRAW main application window according to the new dimensions:

```
Sub AppWindowTop()
With AppWindow
    .WindowState = cdrWindowNormal
    .Top = 10
    .Left = 10
    .Height = 500
    .Width = 600
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_Top')} **Related Topics**

AppWindow.WindowState

Property **WindowState** AS **cdrWindowState**

AppWindow

Description

The **WindowState** property returns or sets the current window state of the CorelDRAW main application window. This window can be minimized, maximized or in a normal state. The application window represents the main CorelDRAW window - it acts as a container for the application.

The **WindowState** property returns a value of cdr WindowState.

Example

The following code example begins with the application window maximized, sets the current window state to minimized, and displays a message that the window has been minimized. The main application window is then restored to a maximized state:

```
Sub AppWindowWindowState()
Dim state As cdrWindowState
state = cdrWindowMaximized    'window should be maximized to begin the procedure
AppWindow.WindowState = cdrWindowMinimized
MsgBox "CorelDRAW is now minimized."
AppWindow.WindowState = state
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_WindowState')} **Related Topics**

AppWindow.ClientWidth

Property **ClientWidth** AS Long

AppWindow

Description

The **ClientWidth** property returns a numerical value that measures the width of a client window within the main application window of CorelDRAW. The application window represents the main CorelDRAW window - it acts as a container for the application.

The **ClientWidth** property returns a Read-Only value and is measured in screen pixels. A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

Example

The following example displays the client width, in screen pixels, in a message box:

```
Sub AppWindowClient()
With AppWindow
    MsgBox .ClientWidth
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_ClientWidth')} **Related Topics**

AppWindow.ClientHeight

Property **ClientHeight** AS Long

AppWindow

Description

The **ClientHeight** property returns a numerical value that measures the height of a client window within the main application window of CorelDRAW. The application window represents the main CorelDRAW window - it acts as a container for the application.

The **ClientHeight** property returns a Read-Only value and is measured in screen pixels. A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

Example

The following example displays the client height, in screen pixels, in a message box:

```
Sub AppWindowClient()
With AppWindow
    MsgBox .ClientHeight
End With
End Sub
```

{button ,AL(`CLS_AppWindow;FNC_ClientHeight')} **Related Topics**

View properties

View Legend

- Application
  - Name
  - OriginX
  - OriginY
  - Page
- Parent
  - UsePage
  - UseZoom
  - Zoom

View methods

View Legend

- Activate
- Delete

View.Delete

Sub **Delete**()

Deletes the view

Member of **View**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_View;FNC_Delete')} **Related Topics**

View

Class **View**

Properties Methods Referenced by

The **View** class defines the characteristics of view objects and describes the look and behavior of the objects through its properties and methods.

A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

You may use several tools to change the way CorelDRAW displays objects in the application window. The View Manager serves two functions. It provides a set of tools for adjusting your view so that you see your drawing the way you want; and it lets you save any view of a specific Drawing Page so that you can revert to it whenever you want.

{button ,AL(`CLS_View')} **Related Topics**

A view is a customized way of looking at a drawing in CorelDRAW. You can customize your views to suit your viewing needs. These views can be saved, deleted, edited, and renamed according to your viewing preferences.

View.Application

Property **Application** AS **Application**

View

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, `CorelScript` and `Application.CorelScript` can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ViewApp()
With ActiveDocument.Views(1)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Application')} **Related Topics**

View.Parent

Property **Parent** AS **Views**

View

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the first view's parent layer in a message box:

```
Sub ViewParent()
With ActiveDocument.Views(1)
    MsgBox Parent.Parent.ActiveLayer.Name
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Parent')} **Related Topics**

View.Name

Property **Name** AS String

View

Description

The **Name** property returns or sets a string value associated with the name of a view in CorelDRAW. A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

A view is given a default name when the view is created (i.e. View 1) but you can change the name of the view at any time in CorelDRAW by using the View Manager.

Example

The following code example sets the name of the first view in the **Views** collection and displays the new name in a message box:

```
Sub ViewName()
With ActiveDocument.Views(1)
    .Name = "My New View"
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Name')} **Related Topics**

View.OriginX

Property **OriginX** AS Double

View

Description

The **OriginX** property returns or sets a value associated with the X coordinate of a view in the **Views** collection of CorelDRAW. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

The X coordinate is the horizontal coordinate of the view, according to the rulers in the Drawing Window. For example, if `Views(1).OriginX = 0` and `Views(1).OriginY = 0`, the view has the coordinate (0, 0) in the top left hand corner of the drawing window, where the horizontal and vertical rulers meet.

Example

The following code example sets the X and Y coordinates of the second view in the **Views** collection of CorelDRAW. The new coordinates are (0, 0), which places the 0 coordinate of the horizontal and vertical rulers in the left hand corner of the Drawing Page:

```
Sub OriginsSet()
With ActiveDocument.Views(2)
    .OriginX = 0
    .OriginY = 0
End With
End Sub
```

{button ,AL(`CLS_View;FNC_OriginX')} **Related Topics**

View.OriginY

Property **OriginY** AS Double

View

Description

The **OriginY** property returns or sets a value associated with the Y coordinate of a view in the **Views** collection of CorelDRAW. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

The Y coordinate is the vertical coordinate of the view, according to the rulers in the Drawing Window. For example, if `Views(1).OriginX = 0` and `Views(1).OriginY = 0`, the view has the coordinate (0, 0) in the top left hand corner of the drawing window, where the horizontal and vertical rulers meet.

Example

The following code example sets the X and Y coordinates of the second view in the **Views** collection of CorelDRAW. The new coordinates are (0, 0), which places the 0 coordinate of the horizontal and vertical rulers in the left hand corner of the Drawing Page:

```
Sub OriginsSet()
With ActiveDocument.Views(2)
    .OriginX = 0
    .OriginY = 0
End With
End Sub
```

{button ,AL(`CLS_View;FNC_OriginY')} **Related Topics**

View.UsePage

Property **UsePage** AS Boolean

View

Description

The **UsePage** property returns or sets a True or False value indicating if the page associated with a view in a multi-page document will take effect when that view becomes active in CorelDRAW. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

If the **UsePage** property is set to True, the page associated with the view appears in the Drawing Window when the view is activated. If the property is set to False, the view's page does not appear when the view is active.

The **UsePage** property has the same function as the page icon beside a view in the View Manager.

Example

The following code example sets the **UsePage** property of the first view in the **Views** collection to True, enabling the page when the view is activated. The first view in the collection is activated with the **Activate** method, and the name of the page associated with the first view in the collection displays in a message box:

```
Sub Page()
With ActiveDocument.Views(1)
    .UsePage = True
    .Activate
    MsgBox .Page.Name
End With
End Sub
```

{button ,AL(`CLS_View;FNC_UsePage')} **Related Topics**

View.Page

Property **Page** AS **Page**

View

Description

The **Page** property returns a value associated with the page of a view in the **Views** collection of CorelDRAW. A view allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

When a custom view is created, it is associated with a specific document page - the custom view is only applicable to that page. In CorelDRAW, you have the option of disabling the page, so that only the zoom properties of a view take affect, not the page properties.

The **Page** property is only applicable if the **UsePage** property is set to True.

Example

The following code example sets the **UsePage** property of the first view in the **Views** collection to True, enabling the page when the view is activated. The first view in the collection is activated with the **Activate** method, and the name of the page associated with the first view in the collection displays in a message box:

```
Sub Page()
With ActiveDocument.Views(1)
    .UsePage = True
    .Activate
    MsgBox .Page.Name
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Page')} **Related Topics**

View.UseZoom

Property **UseZoom** AS Boolean

View

Description

The **UseZoom** property returns or sets a True or False value indicating if the zoom value a view in a multi-page document will take effect when that view becomes active in CorelDRAW. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

Changing the magnification level allows you to zoom in and get a closer look at an area of your drawing, or zoom out and see a wider area of the document. The default zoom level for CorelDRAW is 100%.

If the **UseZoom** property is set to True, the zoom value in a customized view is applied when the view becomes active. If the property is set to False, the view's zoom setting is not applied when the view is active.

This property has the same function as the magnifying glass icon beside a view in the View Manager.

Example

The following code example changes the zoom level in the first view in the **Views** collection of CorelDRAW. The **UseZoom** property is set to True - this enables all **Zoom** properties to be applied in the specified view. The **Zoom** property of the first view is set to 200%:

```
Sub ZoomSet()
With ActiveDocument.Views(1)
    .UseZoom = True
    .Zoom = 200
End With
End Sub
```

{button ,AL(`CLS_View;FNC_UseZoom')} **Related Topics**

View.Zoom

Property **Zoom** AS Long

View

Description

The **Zoom** property returns or sets a user-defined magnification level for a customized view in the **Views** collection of CorelDRAW. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

Changing the magnification level allows you to zoom in and get a closer look at an area of your drawing, or zoom out and see a wider area of the document. The default zoom level for CorelDRAW is 100%.

The **Zoom** property of a custom view is only applicable if the **UseZoom** property of the view is set to True.

Example

The following code example changes the zoom level in the first view in the **Views** collection of CorelDRAW. The **UseZoom** property is set to True - this enables all **Zoom** properties to be applied in the specified view. The **Zoom** property of the first view is set to 200%:

```
With ActiveDocument.Views(1)
    .UseZoom = True
    .Zoom = 200
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Zoom')} **Related Topics**

View.Activate

Sub **Activate**()

View

Description

The **Activate** method opens a view in CorelDRAW and makes that view active in the current document. A view in CorelDRAW allows you to customize how you look at a drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

Example

The following code example activates the second view in the **Views** collection, making it the current view in CorelDRAW:

```
Sub ViewActivate()
With ActiveDocument.Views(2)
    'You may also activate this view by referencing the view's name
    'For example, Views("My Favorite View")
    .Activate
End With
End Sub
```

{button ,AL(`CLS_View;FNC_Activate')} **Related Topics**

Views properties

Views Legend

- Application
- Count
-
- Item
- Parent

Views methods

Views Legend

AddActiveView

Views

Class **Views**

Properties Methods Referenced by

The **Views** class defines the characteristics of Views collection objects and describes the look and behavior of the collection objects through its properties and methods.

A view in CorelDRAW allows you to see your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. The View Manager serves two functions. It provides a set of tools for adjusting your view so that you see your drawing the way you want; and it lets you save any view of a specific Drawing Page so that you can revert to it whenever you want.

You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

{button ,AL(`CLS_Views')} **Related Topics**

Views.Application

Property **Application** AS **Application**

Views

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, `CorelScript` and `Application.CorelScript` can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ViewsApplication()
With ActiveDocument.Views
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Views;FNC_Application')} **Related Topics**

Views.Parent

Property **Parent** AS **Document**

Views

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the caption of the **View** collection's parent object in a message box:

```
Sub ViewsParent()
With ActiveDocument.Views
MsgBox .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Views;FNC_Parent')} **Related Topics**

Views.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **View**

Views

Description

The **Item** property returns a value associated with the index number or name of a view in the **Views** collection of CorelDRAW. A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

`Views(2)` refers to the second view in the **Views** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Views.Item(2)` is the same as `Views(2)` - they both reference the second window in the **Views** collection.

You must reference a view in the collection by passing its index number or name as a parameter:

Parameters	Description
IndexOrName	Index is a preset placeholder for each object in a **Views** collection; it uniquely identifies each member of the collection. Name is a default or user-defined string value that uniquely defines each view in the **Views** collection.

Example

The following code example displays the name of the second customized view in the **Views** collection in a message box:

```
Sub ViewsItem()
With ActiveDocument.Views
    MsgBox .Item(2).Name
End With
End Sub
```

{button ,AL(`CLS_Views;FNC_Item')} **Related Topics**

Views.Count

Property **Count** AS Long

Views

Description

The **Count** property returns the number of views in the **Views** collection of CorelDRAW. A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of views in the **Views** collection in a message box:

```
Sub ViewsCount()
With ActiveDocument.Views
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Views;FNC_Count')} **Related Topics**

Views.AddActiveView

Function **AddActiveView**(ByVal **Name** AS String) AS **View**

Views

Description

The **AddActiveView** method creates a new viewin the **Views** collection in CorelDRAW. A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

The new view is based on the current view settings in the active CorelDRAW document. The view settings of the current view are now the default settings of the new view.

You must name the new view in the collection by passing its new name as a parameter:

Parameters	Description
Name	**Name** is a preset placeholder for each object in a **Views** collection; it uniquely identifies each member of the collection.

Example

The following code example uses the **AddActiveView** method to create a customized view based on the current viewing settings in the active document of CorelDRAW:

```
Sub ViewsAdd()
With ActiveDocument.Views
    .AddActiveView ("My New View")
End With
End Sub
```

{button ,AL(`CLS_Views;FNC_AddActiveView')} **Related Topics**

ActiveView properties

ActiveView Legend

- Application
  - OriginX
  - OriginY
- Parent
  - Type
  - Zoom

ActiveView methods

ActiveView Legend

- SetActualSize
- SetViewPoint
- ToFitAllObjects
- ToFitArea
- ToFitPage
- ToFitPageHeight
- ToFitPageWidth
- ToFitSelection
- ToFitShape
- ToFitShapeRange

ActiveView

Class **ActiveView**

Properties Methods Referenced by

The **ActiveView** class defines the characteristics of ActiveView objects and describes the look and behavior of the objects through its properties and methods. A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

CorelDRAW has five view quality setting types:

Normal view

These settings control how your drawing appears on your monitor. Normal view shows all fills, all objects, and high-resolution bitmaps.

The view-quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Enhanced view

These settings control the way a drawing is displayed on your monitor. Enhanced view uses 2X over-sampling to ensure the best possible display quality.

The view quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Simple Wireframe view

These settings control the way drawings are displayed on your monitor. Simple Wireframe view shows objects as outlines, and hides fills, extrusions, contours, and intermediate blend shapes. Simple Wireframe view also shows monochrome bitmaps. Editing a drawing in Simple Wireframe view is faster because only the object outlines need to be refreshed.

The view-quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Draft view

Shows uniform fills and low-resolution bitmaps. This view displays lenses and fountain fills as colors.

Draft view also displays a unique pattern to represent each fill. The checker board pattern represents two color fills. The two-way arrow pattern represents full-color fills. The hatched line pattern represents the bitmap fill. The PS pattern represents the PostScript fill.

The view quality settings have no effect on the size of a drawing, only on how a drawing is displayed on the monitor.

{button ,AL(`CLS_ActiveView')} **Related Topics**

Normal view

These settings control how your drawing appears on your monitor. Normal view shows all fills, all objects, and high-resolution bitmaps.

The view-quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Enhanced view

These settings control the way a drawing is displayed on your monitor. Enhanced view uses 2X over-sampling to ensure the best possible display quality.

The view quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Wireframe view

A view setting that controls the way drawings are displayed on your computer screen. In Wireframe view, objects display in skeleton form without fills or outlines. Because the screen redraws faster in this view, you may want to use it when you edit complex drawings.

The view-quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Simple Wireframe view

These settings control the way drawings are displayed on your monitor. Simple Wireframe view shows objects as outlines, and hides fills, extrusions, contours, and intermediate blend shapes. Simple Wireframe view also shows monochrome bitmaps. Editing a drawing in Simple Wireframe view is faster because only the object outlines need to be refreshed.

The view-quality settings have no effect on the size of a drawing, only on how the drawing is displayed on the monitor.

Draft view

Shows uniform fills and low-resolution bitmaps. This view displays lenses and fountain fills as colors.

Draft view also displays a unique pattern to represent each fill. The checker board pattern represents two color fills. The two-way arrow pattern represents full-color fills. The hatched line pattern represents the bitmap fill. The PS pattern represents the PostScript fill.

The view quality settings have no effect on the size of a drawing, only on how a drawing is displayed on the monitor.

ActiveView.Application

Property **Application** AS **Application**

ActiveView

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ApplicationActiveView()
With ActiveWindow.ActiveView
    MsgBox "You are using CorelDRAW " & .Version
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_Application')} **Related Topics**

ActiveView.Parent

Property **Parent** AS **Window**

ActiveView

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the caption of the active document in a message box:

```
Sub ActiveViewParent()
With ActiveWindow.ActiveView
    MsgBox .Parent.Caption
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_Parent')} **Related Topics**

ActiveView.Type

Property **Type** AS **cdrViewType**

ActiveView

Description

The **Type** property returns or sets the active view quality in CorelDRAW. A view quality lets you choose how CorelDRAW displays a drawing on the screen. If you have a fast computer or want to see the closest approximation to what a drawing will look like when its' printed, use the Normal or Enhanced view. If you have a slower computer or just want to speed up redrawing of a complex drawing, you may find the Simple Wireframe or Wireframe view the most effective.

The **Type** property returns cdrViewType.

Example

The following code example sets the view type of the current window equal to a value that represents the Enhanced view:

```
Sub DocumentType()
ActiveView.Type = cdrEnhancedView
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_Type')} **Related Topics**

cdrSimpleWireframeView=0
cdrWireframeView=1
cdrDraftView=2
cdrNormalView=3
cdrEnhancedView=4

ActiveView.OriginX

Property **OriginX** AS Double

ActiveView

Description

The **OriginX** property returns or sets the horizontal coordinate of the window's center in CorelDRAW. If you change the x or y coordinate of the active view, you are changing the way the current document is presented on the screen.

Example

The following code example sets the coordinates of the window's center to (0,0) in the document's active view. This places the center of the window in the lower left-hand corner of the page:

```
Sub OriginXOriginY()
With ActiveWindow.ActiveView
    .OriginX = 0
    .OriginY = 0
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_OriginX')} **Related Topics**

ActiveView.OriginY

Property **OriginY** AS Double

ActiveView

Description

The **OriginY** property returns or sets the vertical coordinate of the window's center in CorelDRAW. If you change the x or y coordinate of the active view, you are changing the way the current document is presented on the screen.

Example

The following code example sets the coordinates of the window's center to (0,0) in the document's active view. This places the center of the window in the lower left-hand corner of the page:

```
Sub OriginXOriginY()
With ActiveWindow.ActiveView
    .OriginX = 0
    .OriginY = 0
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_OriginY')} **Related Topics**

ActiveView.Zoom

Property **Zoom** AS Double

ActiveView

Description

The **Zoom** property returns a preset, or sets a user-defined magnification level in the active view of CorelDRAW. Changing the magnification level allows you to zoom in and get a closer look at an area of your drawing, or zoom out and see a wider area of the document. The default zoom level for CorelDRAW is 100%.

Example

The following code example sets the zoom level of the active document to 200%, if the current zoom level is less than 200%:

```
Sub Zoom()
With ActiveWindow.ActiveView
    If .Zoom < 200 Then
    .Zoom = 200
    End If
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_Zoom')} **Related Topics**

ActiveView.ToFitPage

Sub **ToFitPage**()

ActiveView

Description

The **ToFitPage** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit the entire page of the current document. The ToFitPage method ensures that the entire page in a document is visible in CorelDRAW. By default, this sets the zoom level of the active document to 100% and it is the default view property of CorelDRAW.

The **ToFitPage** method is equivalent to the SHIFT+F4 command in CorelDRAW.

Example

The following code example sets the active view of the current document with the ToFitPage method, ensuring that the entire page in the document is visible in the application window:

```
Sub ToFitPage()
ActiveWindow.ActiveView.ToFitPage
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitPage')} **Related Topics**

ActiveView.ToFitPageWidth

Sub **ToFitPageWidth**()

ActiveView

Description

The **ToFitPageWidth** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit the entire page width of the current document into the application window.

The **ToFitPageWidth** method places the left and right borders of the active page to the left and right borders of the application window, adjusting the zoom level so that these page boundaries fit the entire window space.

Example

The following code example adjusts the zoom level in CorelDRAW so that the active view fits the width of the page in the active document. This changes the appearance of the active view in that the entire view space is now occupied by the width of the active page:

```
Sub ToFitPageWidth()
ActiveWindow.ActiveView.ToFitPageWidth
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitPageWidth')}**RelatedTopics**

ActiveView.ToFitPageHeight

Sub **ToFitPageHeight**()

ActiveView

Description

The **ToFitPageHeight** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit the entire page height of the current document into the application window.

The **ToFitPageHeight** method places the top and bottom borders of the active page to the top and bottom borders of the application window, adjusting the zoom level so that these page boundaries fit the entire window space.

Example

The following code example adjusts the zoom level so that the active view fits the height of the page in the active document. This changes the appearance of the active view in that the entire view space is now occupied by the height of the active page:

```
Sub ToFitPageHeight()
ActiveWindow.ActiveView.ToFitPageHeight
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitPageHeight')} **Related Topics**

ActiveView.ToFitShape

Sub **ToFitShape**(ByRef **Shape** AS **Shape**)

ActiveView

Description

The **ToFitShape** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit an entire shape in the current document into the application window. The **ToFitShape** method adjusts the zoom level of the page so that the currently selected shape fills the current view in the application window.

With the **ToFitShape** method, you must specify the shape to be viewed by passing the shape in a parameter:

Parameters	Description
Shape	A **Shape** is any object that can be displayed as several variations of a rectangle or a circle.

Example

The following code example creates a circle in CorelDRAW and adjusts the active view to zoom in on the circle in the active document:

```
Sub ToFitShape()
Dim s As Shape
Set s = ActiveLayer.CreateEllipse2(5, 5, 2)
ActiveWindow.ActiveView.ToFitShape s
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitShape')} **Related Topics**

A parameter is synonymous with argument, a value that is passed to a routine. A parameter defines a characteristic of an object in the visual basic programming environment.

ActiveView.ToFitSelection

Sub **ToFitSelection**()

ActiveView

Description

The **ToFitSelection** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit a selection of objects in the current document into the application window. The **ToFitSelection** method adjusts the zoom level of the page so that the entire current selection of objects fills the current view in the application window.

ToFitSelection method is equivalent to the SHIFT+F2 and F4 commands in CorelDRAW.

Example

The following code example adjusts the zoom level in CorelDRAW so that all selected objects on the page appear in the active view of the current document:

```
Sub ToFitSelection()
ActiveWindow.ActiveView.ToFitSelection
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitSelection')} **Related Topics**

ActiveView.ToFitArea

Sub **ToFitArea**(ByVal **Left** AS Double, ByVal **Top** AS Double, ByVal **Right** AS Double, ByVal **Bottom** AS Double)

ActiveView

Description

The **ToFitArea** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit a specified rectangular area, identified by its coordinates, in the current document into the application window. The **ToFitArea** method defines an area, and adjusts the zoom level of the page so that the defined region fills the current view in the application window.

ToFitArea is the former ToFitRectangle in CorelDRAW.

With the **ToFitArea** method, you must specify the area to be viewed by passing the coordinates of the rectangle in parameters:

Parameters	Description
Left	A coordinate that specifies the distance from the left side of the rectangle to the left side of the page frame in CorelDRAW. The left coordinate is measured in document units.
Top	A coordinate that specifies the distance from the top side of the rectangle to the top of the page frame in CorelDRAW. The top coordinate is measured in document units.
Right	A coordinate that specifies the distance from the right side of the rectangle to the right side of the page frame in CorelDRAW. The right coordinate is measured in document units.
Bottom	A coordinate that specifies the distance from the bottom of the rectangle to the bottom of the page frame in CorelDRAW. The bottom coordinate is measured in document units.

Example

The following code example creates a rectangular area in the active document. This defined area, using the coordinates 0, 0, 5, and 5, uses the ToFitArea method to adjust the zoom level to display the newly defined rectangular page area as the active view:

```
Sub ToFitArea()
ActiveWindow.ActiveView.ToFitArea 0, 0, 5, 5
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitArea')} **Related Topics**

ActiveView.ToFitAllObjects

Sub **ToFitAllObjects**()

ActiveView

Description

The **ToFitAllObjects** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit all the objects on the current page into the application window. The **ToFitAllObjects** method adjusts the zoom level so that the area surrounding all objects on the current page fills up the active view.

Example

The following code example uses the **ToFitAllObjects** method to adjust the zoom level so that all objects on the page fill up the active view of CorelDRAW:

```
Sub ToFitAllObjects()
ActiveWindow.ActiveView.ToFitAllObjects
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitAllObjects')} **Related Topics**

ActiveView.ToFitShapeRange

Sub **ToFitShapeRange**(ByRef **ShapeRange** AS **ShapeRange**)

ActiveView

Description

The **ToFitShapeRange** method adjusts the setting of the active view in CorelDRAW. This method changes the current view to fit an entire collection of shape objects on the current page into the application window. The **ToFitShapeRange** method adjusts the zoom level of the page so that the currently selected shapes fill the current view in the application window.

With the **ToFitShapeRange** method, you must specify the area of shapes to be viewed by passing the ShapeRange in a parameter:

Parameters	Description
ShapeRange	A **ShapeRange** is a collection of shape objects. The ShapeRange defines the area to include in the defined active view of the ToFitShapeRange method.

Example

The following code example creates a rectangle and circle on the current page adds the rectangle and the circle to a shape range, and uses the **ToFitShapeRange** method to adjust the active view and zoom in so that the shapes in the shape range make up the entire view in CorelDRAW:

```
Sub ToFitShapeRange()
Dim s As Shape
Dim sr As New ShapeRange
Set s = ActiveLayer.CreateEllipse2(5, 5, 2)
sr.Add s       'Creates an ellipse and includes it in the defined range
Set s = ActiveLayer.CreateRectangle(0, 0, 2, 2)
sr.Add s       'Creates a rectangle and includes it in the defined range
ActiveWindow.ActiveView.ToFitShapeRange sr
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_ToFitShapeRange')} **Related Topics**

ActiveView.SetViewPoint

Sub **SetViewPoint**(ByVal **X** AS Double, ByVal **Y** AS Double, ByVal Zoom AS Double)

ActiveView

Description

The **SetViewPoint** method sets a viewpoint in the active view of CorelDRAW. A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

The viewpoint represents the center point of what is being viewed through the lens. This point is indicated by an "X" in the Drawing Window.

In order to use the **SetViewPoint** method, you must pass the **X**, **Y**, and **Zoom** values as parameters:

Parameters	Description
X	The **X** parameter identifies the horizontal coordinate of the viewpoint in a lens. The viewpoint is the center point of a lens' view. This value is measured in document units.
Y	The **Y** parameter identifies the vertical coordinate of the viewpoint in a lens. The viewpoint is the center point of a lens' view. This value is measured in document units.
Zoom	The **Zoom** parameter returns a preset, or sets a user-defined magnification level in the lens. Its value is a percentage and is optional in the SetViewPoint method. The default value is 0.

Example

The following code example sets the viewpoint of the active view with the **SetViewPoint** method at the point (2, 2). The zoom is set at 200%:

```
Sub ViewPoint()
With ActiveWindow.ActiveView
    .SetViewPoint 2, 2, 200
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_SetViewPoint')} **Related Topics**

ActiveView.SetActualSize

Sub **SetActualSize**()

ActiveView

Description

The **SetActualSize** method adjusts the current view so that all objects in the view are visible in their actual size. A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

For example, if an ellipse in the active view has a width of 2 inches, the **SetActualSize** method will adjust the current view so the ellipse has a width of 2 inches on the screen.

Example

The following code example sets the current view to reflect the actual size of the object in the view:

```
Sub ViewActual()
With ActiveWindow.ActiveView
.SetActualSize
End With
End Sub
```

{button ,AL(`CLS_ActiveView;FNC_SetActualSize')} **Related Topics**

FontList properties

FontList Legend

- Application

- Count

-
- Item

- Parent

FontList

Class **FontList**

Properties Referenced by

The **FontList** class defines the characteristics of the **FontList** collection in CorelDRAW. The **FontList** collection defines the characteristics of font collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

In CorelDRAW, you can create and edit custom characters. The unique design of a character set is called its typeface. A font is a complete set of characters that share a common typeface -uppercase and lowercase letters, numbers, punctuation marks, and symbols.

A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font. The font list is a collection of font styles in CorelDRAW.

Before you can use a new or modified typeface, you must install it. You can install TrueType fonts using the Windows Control Panel or the Font Navigator Utility. To install Type 1 fonts, you must use the Adobe Type Manager. For information about installing Adobe Type 1 fonts, refer to the documentation provided with Adobe Type Manager.

{button ,AL(`CLS_FontList')} **Related Topics**

FontList.Application

Property **Application** AS Object

FontList

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub FontApp()
With FontList
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_FontList;FNC_Application')} **Related Topics**

FontList.Parent

Property **Parent** AS Object

FontList

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the path of the **FontList** collection's parent object in a message box:

```
Sub FontParent()
With FontList
    MsgBox .Parent.Path
End With
End Sub
```

{button ,AL(`CLS_FontList;FNC_Parent')} **Related Topics**

FontList.Item

Property **Item**(ByVal **Index** AS Long) AS String

FontList

Description

The **Item** property returns a string value associated with the index number of a font object in the **FontList** collection of CorelDRAW. A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font. The font list is a collection of font styles in CorelDRAW.

`FontList(2)` refers to the second font object in the **FontList** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `FontList.Item(2)` is the same as `FontList(2)` - they both reference the second font object in the **FontList** collection.

You must reference a font object in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **FontList** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the name of the fifth font item in the **FontList** collection in a message box:

```
Sub FontItem()
With FontList
    MsgBox .Item(5)
End With
End Sub
```

{button ,AL(`CLS_FontList;FNC_Item')} **Related Topics**

FontList.Count

Property **Count** AS Long

FontList

Description

The **Count** property returns the number of font objects in the **FontList** collection of CorelDRAW. A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font. The font list is a collection of font styles in CorelDRAW.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of fonts in the **FontList** collection in a message box:

```
Sub FontCount()
With FontList
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_FontList;FNC_Count')} **Related Topics**

Workspace properties

Workspace Legend

- Active
- Application
- Default
- Description
- Name
- Parent

Workspace methods

Workspace Legend

Activate

Workspace

Class **Workspace**

Properties Methods Referenced by

The **Workspace** class defines the characteristics of workspace objects and describes the look and behavior of the objects through its properties and methods.

A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

You can create multiple workspaces for specific users or specific tasks and then apply them when required. You can also delete workspaces when they are no longer needed.

You can customize your workspace settings. You can set up your screen the way you want, choose options in the Options dialog box, and then create a custom workspace to save your settings. You can customize the tools and operations that you use most, such as menus and shortcut keys. You can access your custom settings by loading your saved workspace.

{button ,AL(`CLS_Workspace')} **Related Topics**

A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can create multiple workspaces for specific users or specific tasks and then apply them when required. You can also delete workspaces when they are no longer needed.

Workspace.Application

Property **Application** AS Object

Workspace

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub WorkspaceApplication()
With ActiveWorkspace
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Application')} **Related Topics**

Workspace.Parent

Property **Parent** AS **Workspaces**

Workspace

Description

The **Parent** property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the path of the workspace's parent object in a message box:

```
Sub WorkspaceParent()
With ActiveWorkspace
    MsgBox .Parent.Path
End With
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Parent')} **Related Topics**

Workspace.Name

Property **Name** AS String

Workspace

Description

The **Name** property returns the name given to a workspace in CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

The **Name** property returns a String value.

Example

The following code example displays the name of the current workspace in a message box:

```
Sub WorkspaceName()
MsgBox ActiveWorkspace.Name
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Name')} **Related Topics**

Workspace.Description

Property **Description** AS String

Workspace

Description

The **Description** property returns a written description of a workspace in CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

When a workspace is created, you can provide a brief description of the workspace. This is useful if you are using multiple workspaces. The description that you type in the Description Of New Workspace box appears in the list of available workspaces.

The **Description** property returns a Read-Only value.

Example

The following code example displays the description text of the current workspace in a message box:

```
Sub WorkspaceDescription()
MsgBox ActiveWorkspace.Description
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Description')} **Related Topics**

Workspace.Default

Property **Default** AS Boolean

Workspace

Description

The **Default** property returns a True or False value indicating if the current workspace is the default workspace in CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

If the default property is True, the current workspace is the default workspace. The default workspace can be set in the Options dialog box.

The **Default** property returns a Read-Only value.

Example

The following code example checks to see if the current workspace is the default workspace. If it is NOT the default workspace, a message box displays the name of the active workspace in a message box:

```
Sub WorkspaceDefault()
If ActiveWorkspace.Default = False Then
    MsgBox ActiveWorkspace.Name
End If
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Default')} **Related Topics**

Workspace.Activate

Sub **Activate**()

Workspace

Description

The **Activate** method opens a workspace, if the workspace is not currently open in CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

Example

The following code example uses the **Activate** method to open the current workspace if it is not already open in CorelDRAW:

```
Sub WorkspaceActivate()
If Not ActiveWorkspace.Active Then
    ActiveWorkspace.Activate
End If
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Activate')} **Related Topics**

Workspace.Active

Property **Active** AS Boolean

Workspace

Description

The **Active** property returns a True or False value indicating if a specified workspace is currently active in CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

If the **Active** property is True, the workspace is currently active in the application.

The **Active** property returns a Read-Only value.

Example

The following code example displays the name of the current workspace, if the workspace is active, in a message box:

```
Sub WorkspaceActive()
If ActiveWorkspace.Active = True Then
    MsgBox ActiveWorkspace.Name
End If
End Sub
```

{button ,AL(`CLS_Workspace;FNC_Active')} **Related Topics**

Workspaces properties

Workspaces Legend

- Application

- Count

- Item

- Parent

Workspaces

Class **Workspaces**

Properties Referenced by

The **Workspaces** class defines the characteristics of **Workspaces** collection objects and describes the look and behavior of the collection objects through its properties and methods. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW.

A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements. For example, if you are using a low-resolution monitor setting, you can use the preset workspace designed for such a setting.

You can create multiple workspaces for specific users or specific tasks and then apply them when required. You can also delete workspaces when they are no longer needed.

You can customize your workspace settings. You can set up your screen the way you want, choose options in the Options dialog box, and then create a custom workspace to save your settings. You can customize the tools and operations that you use most, such as menus and shortcut keys. You can access your custom settings by loading your saved workspace.

{button ,AL(`CLS_Workspaces')} **Related Topics**

Workspaces.Application

Property **Application** AS Object

Workspaces

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub WorkspacesApplication()
With Workspaces
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Workspaces;FNC_Application')} **Related Topics**

Workspaces.Parent

Property **Parent** AS Object

Workspaces

Description

The **Parent** property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the path of the **Workspaces** collection's parent object in a message box:

```
Sub WorkspacesParent()
With Workspaces
MsgBox .Parent.Path
End With
End Sub
```

{button ,AL(`CLS_Workspaces;FNC_Parent')} **Related Topics**

Workspaces.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **Workspace**

Workspaces

Description

The **Item** property returns a value associated with the index number or name of a workspace object in the Workspaces collection of CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements.

A workspace in CorelDRAW is identified by an index number or a name. You can view the name of each workspace is the Options dialog box.

`Workspaces(2)` refers to the second workspace in the **Workspaces** collection of CorelDRAW. If this workspace is called "Enhanced", `Workspaces(Enhanced)` also refers to the second workspace in the **Workspaces** collection.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Workspaces.Item(2)` is the same as `Workspaces(2)`. They both reference the second workspace in the **Workspaces** collection.

You must reference a workspace in the collection by passing its index number or name as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a Workspaces collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each workspace. Workspace names can be viewed in the Options dialog box.

Example

The following code example sets the second workspace, "Enhanced" as the active workspace in CorelDRAW, if the current view in the document is the Enhanced view. The name of the active workspace, which is "Enhanced", displays in a message box:

```
Sub WorkspacesItem()
Dim MyWorkspace As Workspace
Dim wc As Workspaces
Set MyWorkspace = Workspaces.Item(2)
'the second workspace in the Workspaces collection
'Workspaces.Item(Enhanced) passes the Name parameter
    If ActiveWindow.ActiveView.Type = cdrEnhancedView Then
    'if the current view is the enhanced view
    MyWorkspace.Activate
    'makes the workspace the second workspace
    MsgBox ActiveWorkspace.Name
    End If
End Sub
```

{button ,AL(`CLS_Workspaces;FNC_Item')} **Related Topics**

Workspaces.Count

Property **Count** AS Long

Workspaces

Description

The **Count** property returns the number of workspaces in the **Workspaces** collection of CorelDRAW. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. You can choose from several preset workspaces. Each preset workspace is designed to provide a working environment tailored to your requirements.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of available workspace objects in the **Workspaces** collection of CorelDRAW in a message box:

```
Sub WorkspacesCount()
MsgBox Workspaces.Count
End Sub
```

{button ,AL(`CLS_Workspaces;FNC_Count')} **Related Topics**

Palette properties

Palette Legend

- Application
  Color
- ColorCount
- DuplicatePresent
  Name
- PaletteID
- Parent
- Type

Palette methods

Palette Legend

- AddColor
- Close
- Colors
- GetIndexOfColor
- InsertColor
- RemoveColor
- Save

A swatch is one of a series of solid-colored patches that is used as a sample when selecting color. A printed booklet of color swatches is called a swatch book. Swatch also refers to the colors contained in the Color Palette.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The on-screen color palette is a toolbar that displays a series of color swatches. It is used to select colors for use in CorelDRAW and Corel PHOTO-PAINT. You can display multiple on-screen Color Palettes. They can be docked or left floating in the Application Window.

Palette

Class **Palette**

Properties Methods Referenced by

The **Palette** class defines the characteristics of Palette objects and describes the look and behavior of the objects through its properties and methods.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The on-screen color palette is a toolbar that displays a series of color swatches. It is used to select colors for use in CorelDRAW and Corel PHOTO-PAINT. You can display multiple on-screen Color Palettes. They can be docked or left floating in the Application Window.

The Default on-screen Color Palette is the color palette that you will see in any of the drop down color pickers. You can select any Color Palette to be the Default on-screen Color Palette.

You have the ability to display multiple on-screen Color Palettes, and keep them floating, or dock them to any edge of the Application Window.

There are two types of color palettes from which you can choose colors: fixed color palettes and custom color palettes. On-screen Color Palettes are used to display and select colors from both fixed and custom color palettes.

Fixed color palettes

Fixed color palettes are provided by third-party manufacturers and are most useful when accompanied by a color swatch book. A swatch book is a collection of color samples that shows exactly what each color looks like when printed.

Custom color palettes

Custom color palettes are collections of colors you have chosen to save as a color palette file (.CPL extension).You have the ability to copy a color swatch by dragging a color swatch from any palette into your custom palettes. There is no limit to the number of custom palettes you can create.

When you create a custom palette, the palette is empty and ready for you to choose the colors you want to include in it. You can create a custom palette that includes all the colors from the object that you selected or from the current document.

{button ,AL(`CLS_Palette')} **Related Topics**

Fixed color palettes are provided by third-party manufacturers and are most useful when accompanied by a color swatch book. A swatch book is a collection of color samples that shows exactly what each color looks like when printed.

Palette.Application

Property **Application** AS Object

Palette

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PaletteApp()
With ActivePalette
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Application')} **Related Topics**

Custom color palettes are collections of colors you have chosen to save as a color palette file (.CPL extension).You have the ability to copy a color swatch by dragging a color swatch from any palette into your custom palettes. There is no limit to the number of custom palettes you can create.

When you create a custom palette, the palette is empty and ready for you to choose the colors you want to include in it. You can create a custom palette that includes all the colors from the object that you selected or from the current document.

Palette.Parent

Property **Parent** AS Object

Palette

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the parent object of a color in the active palette of CorelDRAW:

```
Sub PaletteParent(0
With ActivePalette
    MsgBox .Colors.Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Parent')} **Related Topics**

Palette.Name

Property **Name** AS String

Palette

Description

The **Name** property returns or sets a string value associated with the name of a palette object in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The palette name can only be set with the **Name** property if the palette is a custom palette.

Example

The following code example displays the name of the active palette in CorelDRAW:

```
Sub PaletteName()
With ActivePalette
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Name')} **Related Topics**

Palette.Close

Sub **Close**()

Palette

Description

The **Close** method closes a specified color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Example

The following code example closes the active palette in CorelDRAW:

```
Sub PaletteClose()
ActivePalette.Close
End Sub
```

{button ,AL(`CLS_Palette;FNC_Close')} **Related Topics**

Palette.Type

Property **Type** AS **cdrPaletteType**

Palette

Description

The **Type** property returns a value indicating if a color palette is a custom or fixed color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The value returned is cdrPaletteType.

The **Type** property returns a Read-Only value.

Example

The following code example displays the type of the active palette in CorelDRAW:

```
Sub PaletteType()
With ActivePalette
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Type')} **Related Topics**

cdrFixedPalette=0

cdrCustomPalette=1

Palette.Colors

Function **Colors**() AS **Colors**

Palette

Description

The **Colors** property returns or sets a value associated with the Colors collection in a palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Example

The following code example displays the number of colors in the colors collection of the active palette in CorelDRAW:

```
Sub PaletteColors()
With ActivePalette
    MsgBox .Colors.Count
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Colors')} **Related Topics**

Palette.Color

Property **Color**(ByVal **Index** AS Long) AS **Color**

Palette

Description

The **Color** property returns or set a value associated with a specific color in a palette in CorelDRAW. The color is identified by its index number.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

You must reference a color in the palette by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each color object in a Palette; it uniquely identifies each color in the palette object.

Example

The following code example displays the name of the first color in the active palette in a message box:

```
Sub PaletteColorName()
With ActivePalette
    MsgBox .Color(1).Name
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_Color')} **Related Topics**

Palette.AddColor

Sub **AddColor**(ByRef **Color** AS **Color**)

Palette

Description

The **AddColor** method adds a color swatch to a custom palette in CorelDRAW.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

In order to add a color to a custom palette, you must pass the color name as a parameter in the **AddColor** method:

Parameters	Description
Color	Color creates a new color in a palette object in CorelDRAW.

Example

The following code example adds a new color to the active color palette using the **CreateColorEx** method. The new color is added to the active color palette in CorelDRAW:

```
Sub CreatePaletteColor()
ActivePalette.AddColor CreateColorEx(5002, 90, 90, 0, 0)
End Sub
```

{button ,AL(`CLS_Palette;FNC_AddColor')} **Related Topics**

Palette.InsertColor

Sub **InsertColor**(ByVal **Index** AS Long, ByRef **Color** AS **Color**)

Palette

Description

The **InsertColor** method places an existing color into a custom palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

In order to insert a color with the **InsertColor** method, you must past the index and color values as parameters:

Parameters	Description
Index	**Index** is a preset placeholder for each color object in a Palette; it uniquely identifies each color in the palette object.
Color	**Color** creates a new color in a palette object in CorelDRAW.

{button ,AL(`CLS_Palette;FNC_InsertColor')} **Related Topics**

Palette.RemoveColor

Sub **RemoveColor**(ByVal **Index** AS Long)

Palette

Description

The **RemoveColor** method removes a color from a custom palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

In order to remove a color with the **RemoveColor** method, you must pass the color's index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each color object in a Palette; it uniquely identifies each color in the palette object.

Example

The following code example removed the first color from the active palette in CorelDRAW:

```
Sub PaletteRemove()
With ActivePalette
    .RemoveColor(1)
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_RemoveColor')} **Related Topics**

Palette.GetIndexOfColor

Function **GetIndexOfColor**(ByRef **Color** AS **Color**) AS Long

Palette

Description

The **GetIndexOfColor** property returns a value associated with the index number of a color in a palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The **GetIndexOfColor** property returns a Read-Only value.

Parameters	Description
Color	Specifies a color swatch in a palette object in CorelDRAW.

{button ,AL(`CLS_Palette;FNC_GetIndexOfColor')} **Related Topics**

Palette.DuplicatePresent

Property **DuplicatePresent** AS Boolean

Palette

Description

The **DuplicatePresent** property returns a True or False value indicating if a color is repeated in a color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

If the value returned is True, the color is duplicated in the color palette. If the value is False, the color is unique within the palette.

The **DuplicatePresent** property returns a Read-Only value.

Example

The following code example displays a true or false value in a message box, indicating the presence of a duplicated color in the active palette of CorelDRAW:

```
Sub PaletteDuplicate()
With ActivePalette
    MsgBox .DuplicatePresent
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_DuplicatePresent')} **Related Topics**

Palette.ColorCount

Property **ColorCount** AS Long

Palette

Description

The **ColorCount** property returns a value associated with the number of colors in a color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The **ColorCount** property returns a Read-Only value.

Example

The following code example displays the total number of colors found in the active palette of CorelDRAW:

```
Sub PaletteColorCount()
With ActivePalette
    MsgBox .ColorCount
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_ColorCount')} **Related Topics**

Palette.Save

Sub **Save**()

Palette

Description

The **Save** method saves a color palette for future use in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Example

The following code example saves the current color palette in CorelDRAW:

```
Sub PaletteSave()
ActivePalette.Save
End Sub
```

{button ,AL(`CLS_Palette;FNC_Save')} **Related Topics**

Palette.PaletteID

Property **PaletteID** AS **cdrPaletteID**

Palette

Description

The **PaletteID** property returns a value associated with the type of a palette in CorelDRAW. The value returned is cdrPaletteID.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The **PaletteID** returns a Read-Only value.

Example

The following code example displays the ID number of the active palette in a message box. The ID number references a type of palette in CorelDRAW:

```
Sub IDPalette()
With ActivePalette
    MsgBox .PaletteID
End With
End Sub
```

{button ,AL(`CLS_Palette;FNC_PaletteID')} **Related Topics**

cdrTRUMATCH=1

cdrPANTONEProcess=2

cdrPANTONECorel8=3

cdrUniform=7

cdrFOCOLTONE=8

cdrSpectraMaster=9

cdrTOYO=10

cdrDIC=11

cdrPANTONEHexCoated=12

cdrPANTONEHexUncoated=24

cdrLab=13

cdrNetscapeNavigator=14

cdrInternetExplorer=15

cdrPANTONECoated=17

cdrPANTONEUncoated=18

cdrPANTONEMetallic=20

cdrPANTONEPastelCoated=21

cdrPANTONEPastelUncoated=22

cdrHKS=23

cdrCustom=0

Palettes properties

Palettes Legend

- Application
- Count
-
- Item
- Parent

Palettes methods

Palettes Legend

- Create
- CreateFromDocument
- CreateFromSelection
- Open
- OpenFixed

Palettes

Class **Palettes**

Properties Methods Referenced by

The **Palettes** class defines the characteristics of **Palettes** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The on-screen color palette is a toolbar that displays a series of color swatches. It is used to select colors for use in CorelDRAW and Corel PHOTO-PAINT. You can display multiple on-screen Color Palettes. They can be docked or left floating in the Application Window.

The Default on-screen Color Palette is the color palette that you will see in any of the drop down color pickers. You can select any Color Palette to be the Default on-screen Color Palette.

You have the ability to display multiple on-screen Color Palettes, and keep them floating, or dock them to any edge of the Application Window.

There are two types of color palettes from which you can choose colors: fixed color palettes and custom color palettes. On-screen Color Palettes are used to display and select colors from both fixed and custom color palettes.

{button ,AL(`CLS_Palettes')} **Related Topics**

Palettes.Application

Property **Application** AS Object

Palettes

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PalettesApp()
With Palettes
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Application')} **Related Topics**

Palettes.Parent

Property **Parent** AS Object

Palettes

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the parent object of the first palettes' colors in a message box:

```
Sub PalettesParent()
With Palettes.Item(1)
    MsgBox .Colors.Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Parent')} **Related Topics**

Palettes.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **Palette**

Palettes

Description

The **Item** property returns a value associated with the index number of a Palette object in the **Palettes** collection of CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

`Palettes(2)` refers to the second palette object in the **Palettes** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Palettes.Item(2)` is the same as `Palettes(2)` - they both reference the second palette object in the **Palettes** collection.

You must reference a palette in the collection by passing its index number or name as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **Palettes** collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each palette.

Example

The following code example displays the number of colors in the first palette in the Palettes collection in a message box:

```
Sub PalettesItem()
With Palettes.Item(1)
'Palettes(1) may be used here
    MsgBox .Colors.Count
End With
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Item')} **Related Topics**

Palettes.Count

Property **Count** AS Long

Palettes

Description

The **Count** property returns the number of palettes in the **Palettes** collection of CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of open color palettes in the current session of CorelDRAW in a message box:

```
Sub PalettesCount()
MsgBox Palettes.Count
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Count')} **Related Topics**

Palettes.Open

Function **Open**(ByVal **FileName** AS String) AS **Palette**

Palettes

Description

The **Open** method opens an existing color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

In CorelDRAW, you may have several palettes open at the same time. Opening a palette will not close the currently opened palette.

In order to open a color palette with the **Open** method, you must specify the file name of the palette as a parameter:

Parameters	Description
FileName	**FileName** refers to the full path name (with a .cpl extension) of a color palette in CorelDRAW. It is a unique identifier of each palette in the **Palettes** collection. If the Default CMYK Palette is stored on the root of C, its FileName is C:\Default CMYK Palette.cpl

Example

The following code example opens the Gray256 palette in CorelDRAW.

```
Sub PalettesOpen()
Palettes.Open ("C:\ProgramFiles\Corel\Graphics10\Custom\Palettes\Gray256.cpl")
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Open')} **Related Topics**

Palettes.OpenFixed

Function **OpenFixed**(ByVal **PaletteID** AS **cdrPaletteID**) AS **Palette**

Palettes

Description

The **OpenFixed** method opens an existing fixed palette in CorelDRAW, according to its palette ID. . A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

In order to open a fixed palette with the **OpenFixed** method, you must pass the ID of the palette as a parameter.

When passing the **PaletteID** parameter, you have two options. You may pass the numerical ID of the fixed palette or you may pass the CorelDRAW constant name of the fixed palette. For example, `OpenFixed (7)` and `OpenFixed (cdrUniform)` both open the Uniform Colors fixed palette in CorelDRAW.

Parameters	Description
PaletteID	The **PaletteID** property returns a value associated with the type of a palette in CorelDRAW. The value returned is cdrPaletteID.

Example

The following code example opens the Uniform Colors fixed palette in CorelDRAW:

```
Sub OpenFixedPalette()
Palettes.OpenFixed (cdrUniform)
'you may also use Palettes.OpenFixed (7)
End Sub
```

{button ,AL(`CLS_Palettes;FNC_OpenFixed')} **Related Topics**

Palettes.CreateFromDocument

Function **CreateFromDocument**(ByVal **Name** AS String, ByVal Overwrite AS Boolean) AS **Palette**

Palettes

Description

The **CreateFromDocument** method creates a new custom color palette based on the colors used in the current document of CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

If the current document in CorelDRAW uses a range of 15 different colors, the new palette created with the **CreateFromDocument** method is based on the 15 colors.

In order to create a new custom palette with the **CreateFromDocument** method, you must pass a palette name and an overwrite specification as parameters:

Parameters	Description
Name	**Name** is a string value that identifies the new custom palette created from the current document. Since the palette is new, you must give it a new name when you use the **CreateFromDocument** method.
Overwrite	The **Overwrite** parameter allows you to remove the existing palette in CorelDRAW and replace it with the new palette created with the **CreateFromDocument** method. By setting the value to True, you will replace the default palette with the new palette. If set to False, the new palette will co-exist with the default palette. The default value is False.

Example

The following code example creates a new color palette, based on the current document opened in CorelDRAW. The **Overwrite** value is set to False, so the new palette will appear with the default palette in CorelDRAW - it will not replace it. A message box displays the palette type:

```
Sub PaletteCreateDoc()
Dim MyPalette as Palette
Set MyPalette = Palettes.CreateFromPalette ("My New Palette", False)
MsgBox MyPalette.Type
End Sub
```

{button ,AL(`CLS_Palettes;FNC_CreateFromDocument')} **Related Topics**

Palettes.CreateFromSelection

Function **CreateFromSelection**(ByVal **Name** AS String, ByVal Overwrite AS Boolean) AS **Palette**

Palettes

Description

The **CreateFromSelection** method creates a new custom color palette based on the colors used in a selection of objects in the current document of CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

If the current selection in CorelDRAW uses a range of 3 different colors, the new palette created with the **CreateFromSelection** method contains only the 3 colors in the selection.

In order to create a new custom palette with the **CreateFromSelection** method, you must pass a palette name and an overwrite specification as parameters:

Parameters	Description
Name	**Name** is a string value that identifies the new custom palette created from the selection in the current document. Since the palette is new, you must give it a new name when you use the **CreateFromSelction** method.
Overwrite	The **Overwrite** parameter allows you to remove the existing palette in CorelDRAW and replace it with the new palette created with the **CreateFromDocument** method. By setting the value to True, you will replace the default palette with the new palette. If set to False, the new palette will co-exist with the default palette. The default value is False.

Example

The following code example creates a new color palette, based on the selection of objects in the current document opened in CorelDRAW. The **Overwrite** value is set to False, so the new palette will appear with the default palette in CorelDRAW - it will not replace it:

```
Sub PalettesCreateSel()
Palettes.CreateFromSelection ("My New Palette", False)
End Sub
```

{button ,AL(`CLS_Palettes;FNC_CreateFromSelection')} **Related Topics**

Palettes.Create

Function **Create**(ByVal **Name** AS String, ByVal Overwrite AS Boolean) AS **Palette**

Palettes

Description

The **Create** method creates a new empty custom color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper. Once a new palette is created, you can add colors to it according to your needs.

In order to create a new custom palette with the **Create** method, you must pass a palette name and an overwrite specification as parameters:

Parameters	Description
Name	**Name** is a string value that identifies the new custom palette. Since the palette is new, you must give it a new name when you use the **Create** method.
Overwrite	The **Overwrite** parameter allows you to remove the existing palette in CorelDRAW and replace it with the new palette created with the **Create** method. By setting the value to True, you will replace the default palette with the new palette. If set to False, the new palette will co-exist with the default palette. The default value is False.

Example

The following code example creates an empty custom color palette in CorelDRAW. Since the Overwrite parameter is False by default, it does not need to be specified in the expression:

```
Sub PalettesCreate()
Palettes.Create ("My New Palette")
End Sub
```

{button ,AL(`CLS_Palettes;FNC_Create')} **Related Topics**

Color properties

Color Legend

- Application
  - BW
  - CMYCyan
  - CMYKBlack
  - CMYKCyan
  - CMYKMagenta
  - CMYKYellow
  - CMYMagenta
  - CMYYellow
  - Gray
  - HLSHue
  - HLSLightness
  - HLSSaturation
  - HSBBrightness
  - HSBHue
  - HSBSaturation
  - LabComponentA
  - LabComponentB
  - LabLuminance
- Name
- PaletteID
  - PaletteIndex
- Parent
  - RGBBlue
  - RGBGreen
  - RGBRed
  - Tint
- Type
  - YIQChromaI
  - YIQChromaQ
  - YIQLuminanceY

Color methods

Color Legend

- BWAssign
- CMYAssign
- CMYKAssign
- ConvertToBW
- ConvertToCMY
- ConvertToCMYK
- ConvertToFixed
- ConvertToGray
- ConvertToHLS
- ConvertToHSB
- ConvertToLab
- ConvertToRGB
- ConvertToYIQ
- CopyAssign
- CorelScriptAssign
- CorelScriptGetComponent
- FixedAssign
- GrayAssign
- HLSAssign
- HSBAssign
- LabAssign
- RegistrationAssign
- RGBAssign
- UserAssign
- UserAssignEx
- YIQAssign

Color

Class **Color**

Properties Methods Referenced by

The **Color** class defines the characteristics of color objects and describes the look and behavior of the collection objects through its properties and methods.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

There are many different color models that define colors - HSB, RGB, CMYK, and CIE Lab color models. The RGB and CMYK color models are only two of a number of models developed to suit a variety of digital design and desktop publishing applications. It is not necessary to be familiar with all these models, but it is helpful to be familiar with a few of the more widely used ones.

We all see color differently. Color is subjective to the human eye. Each device that interacts with your project's file: the scanner, monitor, and printer may have a different color space. For example, a color that is visible to the human eye may not be reproducible by your printer.

Because there are so many color variations, a precise method for defining each color is required. For example, once you find the perfect shade of light orange, you need to be able to reproduce that color and possibly tell others how to do the same. A color model defines that perfect shade of light orange by breaking it down into precise components that allow you to accurately transmit the information to other people and to the electronic devices you use to create projects.

{button ,AL(`CLS_Color')} **Related Topics**

Color.Application

Property **Application** AS Object

Color

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ColorsApp()
With ActivePalette.Color (1)
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_Application')} **Related Topics**

Color.Parent

Property **Parent** AS Object

Color

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the active color's parent palette in a message box:

```
Sub Parent()
With ActiveShape.Color (1)
MsgBox .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_Parent')} **Related Topics**

Color.Type

Property **Type** AS **cdrColorType**

Color

Description

The **Type** property returns or sets the color type in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The **Type** property returns cdrColorType.

Example

The following code example converts all spot color fills of all objects on the current page to CMYK color model:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type = cdrColorSpot Or s.Fill.UniformColor.Type =
cdrColorPantone Then
                s.Fill.UniformColor.ConvertToCMYK
            End If
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_Type')} **Related Topics**

cdrPantone=1

cdrCMYK=2

cdrCMY=4

cdrRGB=5

cdrHSB=6

cdrHLS=7

cdrBlackAndWhite=8

cdrGray=9

cdrColorPantone=1

cdrColorCMYK=2

cdrColorCMY=4

cdrColorRGB=5

cdrColorHSB=6

cdrColorHLS=7

cdrColorBlackAndWhite=8

cdrColorGray=9

cdrColorYIQ=11

cdrColorLab=12

cdrColorPantoneHex=14

cdrColorRegistration=20

cdrColorSpot=25

cdrColorMixed=99

Color.RGBAssign

Sub **RGBAssign**(ByVal **Red** AS Long, ByVal **Green** AS Long, ByVal **Blue** AS Long)

Color

Description

The **RGBAssign** method assigns the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

Parameters	Description
Red	Sets the level of red in the RGB color model. Values range from 0 to 255.
Green	Sets the level of green in the RGB color model. Values range from 0 to 255.
Blue	Sets the level of blue in the RGB color model. Values range from 0 to 255.

Example

The following code example changes the background color of the current page in the open document to red:

```
Sub PageActive()
ActivePage.Color.RGBAssign 255, 0, 0
'turns the background color red
End Sub
```

{button ,AL(`CLS_Color;FNC_RGBAssign')} **Related Topics**

Color.RGBRed

Property **RGBRed** AS Long

Color

Description

The **RGBRed** property returns or sets the red color value in the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

Example

The following code example deletes all the shape objects with RGB Red fill:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            With s.Fill.UniformColor
                If .Type = cdrColorRGB Then
                    If .RGBRed = 255 And .RGBGreen = 0 And .RGBBlue = 0 Then
                    s.Delete
                    End If
                End If
            End With
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_RGBRed')} **Related Topics**

Color.RGBGreen

Property **RGBGreen** AS Long

Color

Description

The **RGBGreen** property returns or sets the green color value in the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

Example

The following code example deletes all the shape objects with RGB Red fill:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            With s.Fill.UniformColor
                If .Type = cdrColorRGB Then
                    If .RGBRed = 255 And .RGBGreen = 0 And .RGBBlue = 0 Then
                    s.Delete
                    End If
                End If
            End With
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_RGBGreen')} **Related Topics**

Color.RGBBlue

Property **RGBBlue** AS Long

Color

Description

The **RGBBlue** property returns or sets the blue color value in the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

Example

The following code example deletes all the shape objects with RGB Red fill:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            With s.Fill.UniformColor
                If .Type = cdrColorRGB Then
                    If .RGBRed = 255 And .RGBGreen = 0 And .RGBBlue = 0 Then
                    s.Delete
                    End If
                End If
            End With
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_RGBBlue')} **Related Topics**

Color.ConvertToRGB

Sub **ConvertToRGB**()

Color

Description

The **ConvertToRGB** method converts the active color model to the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

CorelDRAW lets you convert an image to the following color modes: Black-and-White, Grayscale, Duotone, RGB, CMYK, or LAB. You can also convert a bitmap to the Paletted color mode.

Example

The following code example uses a separate Color object to take an object's fill color and convert it to RGB, then checks if the color is Red (i.e. its Red component value is greater than 150 and both Blue and Green values are less than 100). If this is the case, the object is filled with black. Otherwise, the objects' fill remains unchanged:

```
Sub Test()
    Dim c As New Color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            c.CopyAssign s.Fill.UniformColor
            c.ConvertToRGB
            If c.RGBRed > 150 And c.RGBGreen < 100 And c.RGBBlue < 100 Then
                s.Fill.UniformColor.RGBAssign 0, 0, 0
            End If
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToRGB')} **Related Topics**

Color.CMYKAssign

Sub **CMYKAssign**(ByVal **Cyan** AS Long, ByVal **Magenta** AS Long, ByVal **Yellow** AS Long, ByVal **Black** AS Long)

Color

Description

The **CMYKAssign** property assigns the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Parameters	Description
Cyan	Sets the level of cyan in the CMYK color model. Values are percentages and range from 0 to 100.
Magenta	Sets the level of magenta in the CMYK color model. Values are percentages and range from 0 to 100.
Yellow	Sets the level of yellow in the CMYK color model. Values are percentages and range from 0 to 100.
Black	Sets the level of black in the CMYK color model. Values are percentages and range from 0 to 100.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.CMYKAssign 100, 0, 100, 0
'Fills the ellipse green
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYKAssign')} **Related Topics**

Color.CMYKCyan

Property **CMYKCyan** AS Long

Color

Description

The **CMYKCyan** property returns or sets the cyan color value in the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Example

The following code example deletes the object if the CMYK color model's cyan component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYKCyan > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYKCyan')} **Related Topics**

Color.CMYKYellow

Property **CMYKYellow** AS Long

Color

Description

The **CMYKYellow** property returns or sets the yellow color value in the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Example

The following code example deletes the object if the CMYK color model's yellow component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYKYellow > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYKYellow')} **Related Topics**

Color.CMYKMagenta

Property **CMYKMagenta** AS Long

Color

Description

The **CMYKYellow** property returns or sets the yellow color value in the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Example

The following code example deletes the object if the CMYK color model's magenta component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYKMagenta > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYKMagenta')} **Related Topics**

Color.CMYKBlack

Property **CMYKBlack** AS Long

Color

Description

The **CMYKBlack** property returns or sets the back color value in the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Example

The following code example deletes the object if the CMYK color model's black component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYKBlack > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYKBlack')} **Related Topics**

Color.ConvertToCMYK

Sub **ConvertToCMYK**()

Color

Description

The **ConvertToCMYK** method converts the active color model to the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Example

The following code example converts all uniform fill colors in the selected shapes to the CMYK color model:

```
Sub Test()
    Dim c As New color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            s.Fill.UniformColor.ConvertToCMYK
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToCMYK')} **Related Topics**

Color.CMYAssign

Sub **CMYAssign**(ByVal **Cyan** AS Long, ByVal **Magenta** AS Long, ByVal **Yellow** AS Long)

Color

Description

The **CMYAssign** method assigns the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Parameters	Description
Cyan	Sets the level of cyan in the CMY color model. Values range from 0 to 255.
Magenta	Sets the level of magenta in the CMY color model. Values range from 0 to 255.
Yellow	Sets the level of yellow in the CMY color model. Values range from 0 to 255.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.CMYAssign 255, 100, 100
'Fills the ellipse based on the CMY color model
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYAssign')} **Related Topics**

Color.CMYCyan

Property **CMYCyan** AS Long

Color

Description

The **CMYCyan** property returns or sets the cyan color in the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Example

The following code example deletes the object if the CMY color model's cyan component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYCyan > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYCyan')} **Related Topics**

Color.CMYMagenta

Property **CMYMagenta** AS Long

Color

Description

The **CMYMagenta** property returns or sets the magenta color in the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Example

The following code example deletes the object if the CMY color model's magenta component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYMagenta > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYMagenta')} **Related Topics**

Color.CMYYellow

Property **CMYYellow** AS Long

Color

Description

The **CMYYellow** property returns or sets the yellow color in the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Example

The following code example deletes the object if the CMY color model's yellow component value is greater than 50:

```
Sub Test()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Fill.UniformColor.CMYYellow > 50 Then
    s.Delete
End If
End Sub
```

{button ,AL(`CLS_Color;FNC_CMYYellow')} **Related Topics**

Color.ConvertToCMY

Sub **ConvertToCMY**()

Color

Description

The **ConvertToCMY** method converts the active color model to the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Example

The following code example converts all uniform fill colors in the selected shapes to the CMY color model:

```
Sub Test()
    Dim c As New color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            s.Fill.UniformColor.ConvertToCMY
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToCMY')} **Related Topics**

Color.HSBAssign

Sub **HSBAssign**(ByVal **Hue** AS Long, ByVal **Saturation** AS Long, ByVal **Brightness** AS Long)

Color

Description

The **HSBAssign** method assigns the HSB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Parameters	Description
Hue	Sets the Hue for the HSB color model. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.
Saturation	Sets the Saturation for the HSB color model. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.
Brightness	Sets the Brightness for the HSB color model. Brightness is the amount of light that is transmitted or reflected from a given pixel. In the HSB color model, brightness is a measure of how much white a color contains. In this case, a brightness value of 0 produces black and a brightness value of 255 produces white.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.HSBAssign 200, 100, 100
'Fills the ellipse based on the HSB color model
End Sub
```

{button ,AL(`CLS_Color;FNC_HSBAssign')} **Related Topics**

Color.HSBHue

Property **HSBHue** AS Long

Color

Description

The **HSBHue** property assigns the Hue value in the HSB color model in CorelDRAW. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Example

The following code example decreases the hue of all the active shape objects on page by half:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type <> cdrColorHSB Then
            s.Fill.UniformColor.ConvertToHSB
            s.Fill.UniformColor.HSBHue = s.Fill.UniformColor.HSBHue / 2
            End If
       End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_HSBHue')} **Related Topics**

Color.HSBSaturation

Property **HSBSaturation** AS Long

Color

Description

The **HSBSaturation** property assigns the Saturation value in the HSB color model in CorelDRAW. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Example

The following code example decreases the saturation of all the active shape objects on page by half:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type <> cdrColorHSB Then
            s.Fill.UniformColor.ConvertToHSB
            s.Fill.UniformColor.HSBSaturation = s.Fill.UniformColor.HSBSaturation / 2
            End If
       End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_HSBSaturation')} **Related Topics**

Color.HSBBrightness

Property **HSBBrightness** AS Long

Color

Description

The **HSBBrightness** property assigns the Saturation value in the HSB color model in CorelDRAW. Brightness is the amount of light that is transmitted or reflected from a given pixel. In the HSB color model, brightness is a measure of how much white a color contains. In this case, a brightness value of 0 produces black and a brightness value of 255 produces white.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Example

The following code example decreases the brightness of all the active shape objects on page by half:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type <> cdrColorHSB Then
            s.Fill.UniformColor.ConvertToHSB
            s.Fill.UniformColor.HSBBrightness = s.Fill.UniformColor.HSBBrightness / 2
            End If
       End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_HSBBrightness')} **Related Topics**

Color.ConvertToHSB

Sub **ConvertToHSB**()

Color

Description

The **ConvertToHSB** method converts the active color model to the HSB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Example

The following code example decreases the saturation of all the active shape objects on page by half:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type <> cdrColorHSB Then
            s.Fill.UniformColor.ConvertToHSB
            s.Fill.UniformColor.HSBSaturation = s.Fill.UniformColor.HSBSaturation / 2
            End If
       End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToHSB')} **Related Topics**

Color.HLSAssign

Sub **HLSAssign**(ByVal **Hue** AS Long, ByVal **Lightness** AS Long, ByVal **Saturation** AS Long)

Color

Description

The **HLSAssign** method assigns the HLS color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Parameters	Description
Hue	Sets the Hue for the HLS color model. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.
Lightness	Sets the Lightness for the HLS color model. Lightness determines the intensity of the color. Values range from 0 to 100.
Saturation	Sets the Saturation for the HLS color model. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.HLSAssign 200, 100, 100
'Fills the ellipse based on the HLS color model
End Sub
```

{button ,AL(`CLS_Color;FNC_HLSAssign')} **Related Topics**

Color.HLSHue

Property **HLSHue** AS Long

Color

Description

The **HLSHue** property assigns the Hue value for the HLS color model in CorelDRAW. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
With ActiveDocument.Selection.Shapes(1).Fill.UniformColor
.HLSAssign 200, 100, 100
'Fills the ellipse based on the HLS color model
MsgBox .HLSHue
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_HLSHue')} **Related Topics**

Color.HLSLightness

Property **HLSLightness** AS Long

Color

Description

The **HLSLightness** property assigns the Lightness value for the HLS color model in CorelDRAW. Lightness determines the intensity of the color. Values range from 0 to 100.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
With ActiveDocument.Selection.Shapes(1).Fill.UniformColor
.HLSAssign 200, 100, 100
'Fills the ellipse based on the HLS color model
MsgBox .HLSLightness
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_HLSLightness')} **Related Topics**

Color.HLSSaturation

Property **HLSSaturation** AS Long

Color

Description

The **HLSSaturation** property assigns the Saturation value for the HLS color model in CorelDRAW. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
With ActiveDocument.Selection.Shapes(1).Fill.UniformColor
.HLSAssign 200, 100, 100
'Fills the ellipse based on the HLS color model
MsgBox .HLSSaturation
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_HLSSaturation')} **Related Topics**

Color.ConvertToHLS

Sub **ConvertToHLS**()

Color

Description

The **ConvertToHLS** method converts the active color model to the HLS color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Example

The following code example converts all uniform fill colors in the selected shapes to the HLS color model:

```
Sub Test()
    Dim c As New color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            s.Fill.UniformColor.ConvertToHLS
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToHLS')} **Related Topics**

Color.BWAssign

Sub **BWAssign**(ByVal **White** AS Boolean)

Color

Description

The **BWAssign** method sets True or False value that assigns the Black and White color model in CorelDRAW. With this method, True means white and False refers to black.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Black and White color model is a 1-bit color mode that stores images as two solid colors - black and white - with no gradations. This color mode is useful for line art and simple graphics.

Parameters	Description
White	Sets the appearance of white in the Black and White color model. This value is a True or False value.

Example

The following code example sets the fill of all uniformly filled objects on the current page to Black & White. Note that it is not necessary to check if the object is Black & White and then set its BW property:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type = cdrColorBlackAndWhite Then
                s.Fill.UniformColor.BW = True
            Else
                s.Fill.UniformColor.BWAssign True
            End If
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_BWAssign')} **Related Topics**

Color.BW

Property **BW** AS Boolean

Color

Description

The **BW** property returns or sets a True or False value that indicates the black and white state in the Black and White color model of CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Black and White color model is a 1-bit color mode that stores images as two solid colors - black and white - with no gradations. This color mode is useful for line art and simple graphics.

Example

The following code example sets the fill of all uniformly filled objects on the current page to Black & White. Note that it is not necessary to check if the object is Black & White and then set its BW property:

```
Sub Test()
    Dim s As Shape
    For Each s In ActivePage.Shapes
        If s.Fill.Type = cdrUniformFill Then
            If s.Fill.UniformColor.Type = cdrColorBlackAndWhite Then
                s.Fill.UniformColor.BW = True
            Else
                s.Fill.UniformColor.BWAssign True
            End If
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_BW')} **Related Topics**

Color.ConvertToBW

Sub **ConvertToBW**()

Color

Description

The **ConvertToBW** method converts the active color model to the Black and White color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Black and White color model is a 1-bit color mode that stores images as two solid colors - black and white - with no gradations. This color mode is useful for line art and simple graphics.

Example

The following code example converts all uniform fill colors in the selected shapes to the Black and White color model:

```
Sub Test()
    Dim c As New color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            s.Fill.UniformColor.ConvertToBW
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToBW')} **Related Topics**

Color.GrayAssign

Sub **GrayAssign**(ByVal **GrayValue** AS Long)

Color

Description

The **GrayAssign** method assigns the Grayscale color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Grayscale color model is a color mode that displays images using 256 shades of gray. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white). In the RGB color mode, a grayscale value corresponds to equal amounts of all RGB colors; in CMYK, a grayscale value corresponds to zero C, M, and Y values, with a positive K value; in HSB, a grayscale value corresponds to zero H and S values, with a positive B value. The Grayscale color mode is based on the Grayscale color model.

Parameters	Description
GrayValue	Sets the level of gray in the Grayscale color model. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white).

Example

The following code example decreases the lightness of the grayscale fill of the object. If the uniform fill color is not grayscale, the code assigns grayscale white to the object fill, prior to changing the lightness:

```
Sub Test()
    Dim s As Shape
    Set s = ActiveSelection.Shapes(1)
    If s.Fill.Type = cdrUniformFill Then
        If s.Fill.UniformColor.Type <> cdrColorGray Then
            s.Fill.UniformColor.GrayAssign 255
            s.Fill.UniformColor.Gray = s.Fill.UniformColor.Gray / 2
        End If
    End If
End Sub
```

{button ,AL(`CLS_Color;FNC_GrayAssign')} **Related Topics**

Color.Gray

Property **Gray** AS Long

Color

Description

The **Gray** property returns or sets the Gray value in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Grayscale color model is a color mode that displays images using 256 shades of gray. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white). In the RGB color mode, a grayscale value corresponds to equal amounts of all RGB colors; in CMYK, a grayscale value corresponds to zero C, M, and Y values, with a positive K value; in HSB, a grayscale value corresponds to zero H and S values, with a positive B value. The Grayscale color mode is based on the Grayscale color model.

Example

The following code example decreases the lightness of the grayscale fill of the object. If the uniform fill color is not grayscale, the code assigns grayscale white to the object fill, prior to changing the lightness:

```
Sub Test()
    Dim s As Shape
    Set s = ActiveSelection.Shapes(1)
    If s.Fill.Type = cdrUniformFill Then
        If s.Fill.UniformColor.Type <> cdrColorGray Then
            s.Fill.UniformColor.GrayAssign 255
            s.Fill.UniformColor.Gray = s.Fill.UniformColor.Gray / 2
        End If
    End If
End Sub
```

{button ,AL(`CLS_Color;FNC_Gray')} **Related Topics**

Color.ConvertToGray

Sub **ConvertToGray**()

Color

Description

The **ConvertToGray** method converts the active color model to the Grayscale color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Grayscale color model is a color mode that displays images using 256 shades of gray. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white). In the RGB color mode, a grayscale value corresponds to equal amounts of all RGB colors; in CMYK, a grayscale value corresponds to zero C, M, and Y values, with a positive K value; in HSB, a grayscale value corresponds to zero H and S values, with a positive B value. The Grayscale color mode is based on the Grayscale color model.

Example

The following code example converts all uniform fill colors in the selected shapes to the Grayscale color model:

```
Sub Test()
    Dim c As New color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            s.Fill.UniformColor.ConvertToGray
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToGray')} **Related Topics**

Color.CorelScriptAssign

Sub **CorelScriptAssign**(ByVal **ColorModel** AS Long, ByVal **V1** AS Long, ByVal V2 AS Long, ByVal V3 AS Long, ByVal V4 AS Long, ByVal V5 AS Long, ByVal V6 AS Long, ByVal V7 AS Long)

Color

Description

The **CorelScriptAssign** method assigns a color object based on Corel SCRIPT color specification values in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Parameters	Description
ColorModel	The **ColorModel** parameter identifies the color model used to create the new color. This parameter specifies the numeric variable that is assigned to the color model. A color model is a simple color chart that defines the range of colors displayed in a color mode.
V1	The **V1** parameter specifies the numeric variable that is assigned to the first color component of the selected color model. For example, cyan is the first color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V2	The V2 parameter specifies the numeric variable that is assigned to the second color component of the selected color model. This value is optional and the default value is 0.
V3	The **V3** parameter specifies the numeric variable that is assigned to the third color component of the selected color model. This value is optional and the default value is 0.
V4	The **V4** parameter specifies the numeric variable that is assigned to the fourth color component of the selected color model. This value is optional and the default value is 0.
V5	The **V5** parameter specifies the numeric variable that is assigned to the fifth color component of the selected color model. This value is optional and the default value is 0.
V6	The **V6** parameter specifies the numeric variable that is assigned to the sixth color component of the selected color model. This value is optional and the default value is 0.
V7	The **V7** parameter specifies the numeric variable that is assigned to the seventh color component of the selected color model. This value is optional and the default value is 0.

Example

The following code example sets the background color of the current page to RGB Red:

```
Sub Test()
    ActiveDocument.ActivePage.Color.CorelScriptAssign 5005, 255, 0, 0
End Sub
```

{button ,AL(`CLS_Color;FNC_CorelScriptAssign')} **Related Topics**

Color.CorelScriptGetComponent

Sub **CorelScriptGetComponent**(ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef V2 AS Long, ByRef V3 AS Long, ByRef V4 AS Long, ByRef V5 AS Long, ByRef V6 AS Long, ByRef V7 AS Long)

Color

Description

The **CorelScriptGetComponent** method gets Corel SCRIPT values from a color object for use in Corel SCRIPT. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Parameters	Description
ColorModel	The **ColorModel** parameter identifies the color model used to create the new color. This parameter specifies the numeric variable that is assigned to the color model. A color model is a simple color chart that defines the range of colors displayed in a color mode.
V1	The **V1** parameter specifies the numeric variable that is assigned to the first color component of the selected color model. For example, cyan is the first color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V2	The V2 parameter specifies the numeric variable that is assigned to the second color component of the selected color model. This value is optional and the default value is 0.
V3	The **V3** parameter specifies the numeric variable that is assigned to the third color component of the selected color model. This value is optional and the default value is 0.
V4	The **V4** parameter specifies the numeric variable that is assigned to the fourth color component of the selected color model. This value is optional and the default value is 0.
V5	The **V5** parameter specifies the numeric variable that is assigned to the fifth color component of the selected color model. This value is optional and the default value is 0.
V6	The **V6** parameter specifies the numeric variable that is assigned to the sixth color component of the selected color model. This value is optional and the default value is 0.
V7	The **V7** parameter specifies the numeric variable that is assigned to the seventh color component of the selected color model. This value is optional and the default value is 0.

{button ,AL(`CLS_Color;FNC_CorelScriptGetComponent')} **Related Topics**

Color.UserAssign

Sub **UserAssign**()

Color

Description

The **UserAssign** method brings up the Color dialog box, allowing the user to assign a color in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Example

The following code example creates a new color object, then shows the color dialog and prompts the user to select a color. The code then applies the color to all the object fills in the selection:

```
Sub Test()
Dim c As New color
Dim s As Shape
c.UserAssign
  For Each s In ActiveSelection.Shapes
    s.Fill.ApplyUniformFill c
  Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_UserAssign')} **Related Topics**

Color.CopyAssign

Sub **CopyAssign**(ByRef **Color** AS **Color**)

Color

Description

The **CopyAssign** method copies the color model properties from a color object and applies them to the active color in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Parameters	Description
Color	The **Color** parameter identifies the color who's color model properties are copied with the CopyAssign method.

Example

The following code example uses a separate Color object to take an object's fill color and convert it to RGB, then checks if the color is Red (i.e. its Red component value is greater than 150 and both Blue and Green values are less than 100). If this is the case, the object is filled with black. Otherwise, the objects' fill remains unchanged:

```
Sub Test()
    Dim c As New Color
    Dim s As Shape
    For Each s In ActiveSelection.Shapes
        If s.Fill.Type = cdrUniformFill Then
            c.CopyAssign s.Fill.UniformColor
            c.ConvertToRGB
            If c.RGBRed > 150 And c.RGBGreen < 100 And c.RGBBlue < 100 Then
                s.Fill.UniformColor.RGBAssign 0, 0, 0
            End If
        End If
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_CopyAssign')} **Related Topics**

Color.Name

Property **Name**(ByVal Components AS Boolean) AS String

Color

Description

The **Name** property returns a string value that identifies a color in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

This property returns a Read-Only value.

Parameters	Description
Components	The **Components** parameter sets a True or False value that indicates whether or not to return the color's components when using the Name property. This value is optional and the default value is False.

Example

The following code example displays the name of the color and its color component values:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
MsgBox "Fill Color Name: " & .Name & vbCr & _
"Components: " & .Name (True)
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_Name')} **Related Topics**

Color.YIQAssign

Sub **YIQAssign**(ByVal **Y** AS Long, ByVal **I** AS Long, ByVal **Q** AS Long)

Color

Description

The **YIQAssign** method assigns the YIQ color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Parameters	Description
Y	Sets the luminance, or brightness, value in the YIQ color model. Values range from 0 to 255.
I	Sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.
Q	Sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.

Example

The following code example sets the fill of the object to YIQ purple:

```
Sub Test()
    ActiveSelection.Shapes(1).Fill.UniformColor.YIQAssign 100, 255, 255
End Sub
```

{button ,AL(`CLS_Color;FNC_YIQAssign')} **Related Topics**

Color.YIQLuminanceY

Property **YIQLuminanceY** AS Long

Color

Description

The **YIQLuminanceY** returns or sets the Y color value in the YIQ color model in CorelDRAW. The Y value sets the luminance, or brightness, in the YIQ color model. Values range from 0 to 255.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Example

The following code example sets the fill of the object to YIQ purple and displays the YIQLuminanceY value in a message box:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
.YIQAssign 100, 255, 255
MsgBox .YIQLuminanceY
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_YIQLuminanceY')} **Related Topics**

Color.YIQChromaI

Property **YIQChromaI** AS Long

Color

Description

The **YIQChromaII** returns or sets the I color value in the YIQ color model in CorelDRAW. The I value sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Example

The following code example sets the fill of the object to YIQ purple and displays the YIQChromaII value in a message box:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
.YIQAssign 100, 255, 255
MsgBox .YIQChromaII
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_YIQChromaI')} **Related Topics**

Color.YIQChromaQ

Property **YIQChromaQ** AS Long

Color

Description

The **YIQChromaIQ** returns or sets the Q color value in the YIQ color model in CorelDRAW. The Q value sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Example

The following code example sets the fill of the object to YIQ purple and displays the YIQChromalQ value in a message box:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
.YIQAssign 100, 255, 255
MsgBox .YIQChromalQ
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_YIQChromaQ')} **Related Topics**

Color.ConvertToYIQ

Sub **ConvertToYIQ**()

Color

Description

The **ConvertToYIQ** method converts the active color model to the YIQ color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Example

The following code example converts all uniform fill colors in the selected shapes to the YIQ color model:

```
Sub Test()
Dim c As New color
Dim s As Shape
For Each s In ActiveSelection.Shapes
  If s.Fill.Type = cdrUniformFill Then
    s.Fill.UniformColor.ConvertToYIQ
  End If
Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToYIQ')} **Related Topics**

Color.LabAssign

Sub **LabAssign**(ByVal **L** AS Long, ByVal **A** AS Long, ByVal **B** AS Long)

Color

Description

The **LabAssign** method assigns the LAB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Parameters	Description
L	Sets the Luminance, or brightness, value in the LAB color model. Values range from 0 to 255.
A	Sets one of two chromatic components in the LAB color model. "A" refers to a range of colors between green and red.
B	Sets one of two chromatic components in the LAB color model. "B" refers to a range of colors between blue and yellow.

Example

The following code example fills all shape objects with the Lab gray color:

```
Sub Test()
Dim s As Shape
For Each s In ActivePage.Shapes
   If s.Fill.Type = cdrUniformFill Then
     s.Fill.UniformColor.LabAssign 128, 0, 0
   End If
Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_LabAssign')} **Related Topics**

Color.LabLuminance

Property **LabLuminance** AS Long

Color

Description

The **LabLuminance** property returns or sets the luminance level in the LAB color model in CorelDRAW. This property sets the Luminance, or brightness, value in the LAB color model. Values range from 0 to 255.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Example

The following code example fills all shape objects with the Lab gray color and displays the luminance value in a message box:

```
Sub Test()
Dim s As Shape
For Each s In ActivePage.Shapes
    If s.Fill.Type = cdrUniformFill Then
      s.Fill.UniformColor.LabAssign 128, 0, 0
    End If
Next s
MsgBox s.Fill.UniformColor.LabLuminance
End Sub
```

{button ,AL(`CLS_Color;FNC_LabLuminance')} **Related Topics**

Color.LabComponentA

Property **LabComponentA** AS Long

Color

Description

The **LabComponentA** property returns or sets the A component in the LAB color model in CorelDRAW. This property sets one of two chromatic components in the LAB color model. "A" refers to a range of colors between green and red.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Example

The following code example fills all shape objects with the Lab gray color and displays the lab component A value in a message box:

```
Sub Test()
Dim s As Shape
For Each s In ActivePage.Shapes
    If s.Fill.Type = cdrUniformFill Then
      s.Fill.UniformColor.LabAssign 128, 0, 0
    End If
Next s
MsgBox s.Fill.UniformColor.LabComponentA
End Sub
```

{button ,AL(`CLS_Color;FNC_LabComponentA')} **Related Topics**

Color.LabComponentB

Property **LabComponentB** AS Long

Color

Description

The **LabComponentB** property returns or sets the B component in the LAB color model in CorelDRAW. This property sets one of two chromatic components in the LAB color model. "B" refers to a range of colors between blue and yellow.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Example

The following code example fills all shape objects with the Lab gray color and displays the lab component B value in a message box:

```
Sub Test()
Dim s As Shape
For Each s In ActivePage.Shapes
    If s.Fill.Type = cdrUniformFill Then
      s.Fill.UniformColor.LabAssign 128, 0, 0
    End If
Next s
MsgBox s.Fill.UniformColor.LabComponentB
End Sub
```

{button ,AL(`CLS_Color;FNC_LabComponentB')} **Related Topics**

Color.ConvertToLab

Sub **ConvertToLab**()

Color

Description

The **ConvertToLab** method converts the active color model to the LAB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Example

The following code example converts all uniform fill colors in the selected shapes to the LAB color model:

```
Sub Test()
Dim c As New color
Dim s As Shape
For Each s In ActiveSelection.Shapes
    If s.Fill.Type = cdrUniformFill Then
      s.Fill.UniformColor.ConvertToLab
    End If
Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToLab')} **Related Topics**

Color.RegistrationAssign

Sub **RegistrationAssign**()

Color

Description

The **RegistrationAssign** method assigns the Registration color model that allows color trapping in CorelDRAW. Color trapping is necessary to compensate for poor color registration that occurs when the printing plates used to print each color, called color separations, are not aligned perfectly. Poor registration causes unintentional white slivers to appear between adjoining colors. Trapping is accomplished by intentionally overlapping colors so that minor problems with alignment are not noticed.

Color trapping is achieved by overprinting. Usually, portions of an object that are obscured by another object are not printed. However, if the top object is set to overprint, the obscured portions of any underlying objects print anyway, causing an overlap. This makes white gaps between different colors unlikely. Overprinting works best when the top color is much darker than the underlying color; otherwise, an undesirable third color may result (for example, red over yellow may result in an orange object).

Example

The following code example builds crop marks around selected objects and sets their outline color to the registration color:

```
Sub Test()
    Const Offset As Double = 1 ' Crop mark offset
    Const Length As Double = 5 ' Crop mark length
    Dim s As Shape
    Dim r As New ShapeRange
    Dim lyr As Layer
    Dim x1 As Double, y1 As Double, x2 As Double, y2 As Double
    Set s = ActiveSelection
    Set lyr = ActiveDocument.ActiveLayer
    ActiveDocument.Unit = cdrMillimeter
    ActiveDocument.ReferencePoint = cdrTopLeft
    x1 = s.PositionX
    y1 = s.PositionY
    ActiveDocument.ReferencePoint = cdrBottomRight
    x2 = s.PositionX
    y2 = s.PositionY
    r.Add lyr.CreateLineSegment(x1 - Length - Offset, y1, x1 - Offset, y1)
    r.Add lyr.CreateLineSegment(x1, y1 + Length + Offset, x1, y1 + Offset)
    r.Add lyr.CreateLineSegment(x2 + Offset, y1, x2 + Length + Offset, y1)
    r.Add lyr.CreateLineSegment(x2, y1 + Length + Offset, x2, y1 + Offset)
    r.Add lyr.CreateLineSegment(x1 - Length - Offset, y2, x1 - Offset, y2)
    r.Add lyr.CreateLineSegment(x1, y2 - Length - Offset, x1, y2 - Offset)
    r.Add lyr.CreateLineSegment(x2 + Offset, y2, x2 + Length + Offset, y2)
    r.Add lyr.CreateLineSegment(x2, y2 - Length - Offset, x2, y2 - Offset)
    For Each s In r
        s.Outline.Width = 0.1
        s.Outline.Color.RegistrationAssign
    Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_RegistrationAssign')} **Related Topics**

Color.FixedAssign

Sub **FixedAssign**(ByVal **PaletteID** AS **cdrPaletteID**, ByVal **PaletteIndex** AS Long, ByVal Tint AS Long)

Color

Description

The **FixedAssign** method assigns a fixed palette color in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Parameters	Description
PaletteID	Sets the Palette ID that uniquely identifies which color palette to use in CorelDRAW.
PaletteIndex	Sets the color to use by referencing the index number of a color within a palette.
Tint	Sets the tint of the palette color. Tints are lighter shades of a spot color that are created by changing the percentage tint value. This value is optional and the default value is 100.

Example

The following code example fills the object with a 50% PANTONE Process Yellow CVC color:

```
Sub Test()
    ActiveSelection.Shapes(1).Fill.UniformColor.FixedAssign cdrPANTONECoated, 1, 50
End Sub
```

{button ,AL(`CLS_Color;FNC_FixedAssign')} **Related Topics**

Color.PaletteID

Property **PaletteID** AS **cdrPaletteID**

Color

Description

The **PaletteID** property returns a value associated with the type of a palette in CorelDRAW. The value returned is cdrPaletteID.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

The PaletteID returns a Read-Only value.

Example

The following code example displays the palette ID number of the end color used in the fountain fill effect in the active shape of CorelDRAW:

```
Sub FillFountain()
With ActiveShape.Fill.Fountain
    MsgBox .EndColor.PaletteID
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_PaletteID')} **Related Topics**

Color.PaletteIndex

Property **PaletteIndex** AS Long

Color

Description

The **PaletteIndex** property returns or sets the index value of a color in a palette in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Example

The following code example fills the object with a 50% PANTONE Process Yellow CVC color and displays the palette index in a message box:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
.FixedAssign cdrPANTONECoated, 1, 50
MsgBox .PaletteIndex
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_PaletteIndex')} **Related Topics**

Color.Tint

Property **Tint** AS Long

Color

Description

The **Tint** property returns or sets the tint level in a fixed palette's color in CorelDRAW. Tints are lighter shades of a spot color that are created by changing the percentage tint value.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Example

The following code example fills the object with a 50% PANTONE Process Yellow CVC color and displays the tint value in a message box:

```
Sub Test()
With ActiveSelection.Shapes(1).Fill.UniformColor
.FixedAssign cdrPANTONECoated, 1, 50
MsgBox .Tint
End With
End Sub
```

{button ,AL(`CLS_Color;FNC_Tint')} **Related Topics**

Color.ConvertToFixed

Sub **ConvertToFixed**(ByVal **PaletteID** AS **cdrPaletteID**)

Color

Description

The **ConvertToFixed** method converts the active color model to a color in the fixed palette of CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

Parameters	Description
PaletteID	Sets the fixed palette in CorelDRAW. This parameter returns cdrPaletteID.

Example

The following code example converts all uniform fill colors in the selected shapes to the Netscape web-safe palette:

```
Sub Test()
Dim c As New color
Dim s As Shape
For Each s In ActiveSelection.Shapes
   If s.Fill.Type = cdrUniformFill Then
     s.Fill.UniformColor.ConvertToFixed cdrNetscapeNavigator
   End If
Next s
End Sub
```

{button ,AL(`CLS_Color;FNC_ConvertToFixed')} **Related Topics**

Color.UserAssignEx

Function **UserAssignEx**() AS Boolean

Color

Description

The **UserAssignEx** method sets a True or False value that brings up the Color dialog box, allowing the user to select a color in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

{button ,AL(`CLS_Color;FNC_UserAssignEx')} **Related Topics**

Colors properties

Colors Legend

- Application
- Count
-
- Item
- Parent

A great deal of color research has been accomplished in order to acquire a color model that is device independent and repeatable. In 1931 La Commision Internationale de L'Eclairage (CIE) defined a device-independent color model, based on how the human eye perceives color. The CIE Lab model incorporates the theory that a color cannot be both green and red at the same time nor can it be yellow and blue at the same time. As such, single values are used to describe the green/red and blue/yellow components of any color. Lab stands for the three values this model uses to define colora lightness value (L) which can range from 0 to 100 and two chromaticity ranges: green to red (a) and blue to yellow (b). The two chromaticity values can range from +120 to -120. Lab (sometimes called L*a*b*) provides a system for defining color that bases color values on widely accepted standards rather than on individual color-producing devices.

The millions of colors your monitor produces can all be described as amounts of red, green, and blue. These three color components form the basis for the RGB (Red, Green, and Blue) color model. Each of the three colors is assigned a numeric value between 0 and 255. The RGB model is based on colors of light, and higher RGB values correspond to the presence of greater quantities of white light. Consequently, higher RGB values result in lighter colors. When all three color components are at the maximum value, the resulting color is white light. Because the RGB model creates colors by adding light, it is called an additive color model. Monitors and scanners can employ the additive color model because they emit light. They emit particles of red, green, and blue light and create the illusion of millions of different colors.

One of the limitations of the RGB model is that it is device dependent. This means that not only are there color variations between monitors and scanners by different manufacturers but there are color variations between identical devices from the same manufacturer. All monitors drift over time and display colors differently making it imperative to regularly calibrate your monitor and the other electronic devices you use to create your projects. The RGB model cannot be a color standard because its color results are not 100 percent repeatable.

Without any light or a viewer, objects all around us are colorless. Color only occurs in our minds after our visual sensory system has seen the wavelengths that give objects their color. Based on how people perceive color, the HSB color model defines color in three attributes:

- Hue (H)
- Saturation (S)
- Brightness (B)

Hue (H) is the name we give a color in everyday language. Hues form the Color Wheel. The hue of a lemon is yellow, that of a strawberry is red. Saturation (S) refers to vividness of the color or how much color concentration does the object contain. The figurine does not contain very much yellow when compared to the yellow saturation of lemon. Colors can be separated into bright or dark colors when their Brightness (B) is compared. Brightness refers to adding or removing whiteness from a color. The mask is bright and lighter then the dark yellow lemon.

Colors

Class **Colors**

Properties Referenced by

The **Colors** class defines the characteristics of **Colors** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

There are many different color models that define colors - HSB, RGB, CMYK, and CIE Lab color models. The RGB and CMYK color models are only two of a number of models developed to suit a variety of digital design and desktop publishing applications. It is not necessary to be familiar with all these models, but it is helpful to be familiar with a few of the more widely used ones.

We all see color differently. Color is subjective to the human eye. Each device that interacts with your project's file: the scanner, monitor, and printer may have a different color space. For example, a color that is visible to the human eye may not be reproducible by your printer.

Because there are so many color variations, a precise method for defining each color is required. For example, once you find the perfect shade of light orange, you need to be able to reproduce that color and possibly tell others how to do the same. A color model defines that perfect shade of light orange by breaking it down into precise components that allow you to accurately transmit the information to other people and to the electronic devices you use to create projects.

{button ,AL(`CLS_Colors')} **Related Topics**

Colors.Application

Property **Application** AS Object

Colors

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ColorsApp()
With ActivePalette.Colors
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Colors;FNC_Application')} **Related Topics**

Colors.Parent

Property **Parent** AS **Palette**

Colors

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the **Colors** collection's parent palette in a message box:

```
Sub ColorsParent()
With ActivePalette.Colors
    MsgBox .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_Colors;FNC_Parent')} **Related Topics**

Colors.Item

Property **Item**(ByVal **Index** AS Long) AS **Color**

Colors

Description

The **Item** property returns a value associated with the index number of a color in the **Colors** collection of CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

`Colors(2)` refers to the second color in the **Colors** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Colors.Item(2)` is the same as `Colors(2)` - they both reference the second color in the **Colors** collection.

You must reference a color in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each color in a **Colors** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the color type of the **Colors** collection's third color in a message box:

```
Sub ColorsItem()
With ActivePalette.Colors
    MsgBox .Item(3).Type
End With
End Sub
```

{button ,AL(`CLS_Colors;FNC_Item')} **Related Topics**

Colors.Count

Property **Count** AS Long

Colors

Description

The **Count** property returns the number of colors in the **Colors** collection of CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of colors in the active palette's **Colors** collection in a message box:

```
Sub ColorsCount()
With ActivePalette.Colors
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Colors;FNC_Count')} **Related Topics**

Outline properties

Outline Legend

- BehindFill
- Color
- EndArrow
- LineCaps
- LineJoin
- NibAngle
- NibStretch
- ScaleWithShape
- StartArrow
- Style
- Type
- Width

Outline methods

Outline Legend

- ConvertToObject
- SetProperties

Outline

Class **Outline**

Properties Methods Referenced by

The **Outline** class defines the characteristics of outline objects and describes the look and behavior of the objects through its properties and methods. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

You can convert an outline to an object, which lets you apply a fill to an outline. When you use thick outlines, the fill can continue through to the outer edge of the object, including the outline. When an outline is converted, it acquires all the properties of any object.

You can change the size, shape, and color of outlines. Lines or objects with open paths can have ends that are rounded, square, cropped, or tipped with arrowheads and other line-ending shapes. Objects with closed paths (e.g., squares and polygons) do not have end-points, but you can choose from pointed, rounded, or truncated corners.

{button ,AL(`CLS_Outline')} **Related Topics**

Outline.Width

Property **Width** AS Double

Outline

Description

The **Width** property returns or sets a numerical value that indicates the width of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

The width of an outline determines the thickness of the line in points. Changing the thickness of an object's outline changes the appearance of the object.

A number of preset outline widths are also available from the Outline Tool flyout. Options include: Hairline, 1/2 Point, 2 Point (Thin), 8 Point (Medium), 16 Point (Medium-Thick), and 24 Point (Thick).

Example

The following code example creates a new outline style and applies the new style to a line segment that is created on the active layer of the current document in CorelDRAW:

```
Sub StylesAdd()
Dim NewShape as Shape
Dim NewStyle as Style
Set NewStyle = OutlineStyles.Add      'creates a new style
NewStyle.DashCount = 1        '1 dash/gap pair
NewStyle.DashLength(1) = 20           'sets dash length
NewStyle.GapLength(1) = 1             'sets gap length
Set NewShape = ActiveLayer.CreateLineSegment (0, 0, 5, 5)
NewShape.Outline.Width = .1           'sets width of line
NewShape.Outline.Style = NewStyle     'sets outline style to newly created style
End Sub
```

{button ,AL(`CLS_Outline;FNC_Width')} **Related Topics**

Outline.Color

Property **Color** AS **Color**

Outline

Description

The **Color** property returns or sets the color of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

You can change the color of an outline to create a contrast to the object's fill color. Applying an outline using the Outline Color dialog box lets you exercise more control over the outline color. Applying an outline using the Outline Pen dialog box lets you set the outline thickness, style and corner shape.

To color an outline quickly you can use the on-screen Color Palette, enter the color components of a color on the Property Bar, or drag a color from the Color Palette to the object's edge. As the mouse pointer moves over the object, it changes shape to show where the color will be applied.

Example

The following code example displays the name of the outline color in the active shape of CorelDRAW:

```
Sub Color()
MsgBox ActiveShape.Outline.Color.Name
End Sub
```

{button ,AL(`CLS_Outline;FNC_Color')} **Related Topics**

Outline.ConvertToObject

Function **ConvertToObject**() AS **Shape**

Outline

Description

The **ConvertToObject** method converts an object's outline into a separate object in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

You can convert an outline to an object, which lets you apply a fill to an outline. When you use thick outlines, the fill can continue through to the outer edge of the object, including the outline. When an outline is converted, it acquires all the properties of any object.

You can only convert objects that have an outline. If the Behind Fill option is enabled, the outline object appears behind the original object.

Example

The following code example converts the outline of the active shape into a separate object in CorelDRAW:

```
Sub Convert()
ActiveShape.Outline.ConvertToObject
End Sub
```

{button ,AL(`CLS_Outline;FNC_ConvertToObject')} **Related Topics**

Outline.Type

Property **Type** AS **cdrOutlineType**

Outline

Description

The **Type** property returns or sets the outline type of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

The **Type** property returns a value of cdrOutlineType.

Example

The following code example applies predefined outline styles to selected objects. The style that is applied depends on the type of shape selected - if the shape is a rectangle, the first preset outline style is applied to the outline of the rectangle. Only shapes that have an outline are included the selection:

```
Sub StylesOutline()
Dim s As Shape
For Each s In ActiveSelection.Shapes
    If s.Outline.Type = cdrOutline Then
    Select Case s.Type
    Case cdrRectangleShape
        s.Outline.Style = OutlineStyles(1)   'first preset outline style
    Case cdrEllipseShape
        s.Outline.Style = OutlineStyles(2)   'second preset outline style
    Case cdrPolygonShape
        s.Outline.Style = OutlineStyles(3)   'third preset outline style
    End Select
    End If
Next s
End Sub
```

{button ,AL(`CLS_Outline;FNC_Type')} **Related Topics**

Outline.Style

Property **Style** AS **OutlineStyle**

Outline

Description

The **Style** property returns or sets the outline style of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

CorelDRAW comes with many different line styles. Line styles are preset lines that have different attributes, such as dotted lines, or dashed lines. Applying a line style does not change the shape of the line or the amount of space it occupies. You can edit an existing line style to meet your needs.

Example

The following code example applies predefined outline styles to selected objects. The style that is applied depends on the type of shape selected - if the shape is a rectangle, the first preset outline style is applied to the outline of the rectangle. Only shapes that have an outline are included the selection:

```
Sub StylesOutline()
Dim s As Shape
For Each s In ActiveSelection.Shapes
    If s.Outline.Type = cdrOutline Then
    Select Case s.Type
    Case cdrRectangleShape
        s.Outline.Style = OutlineStyles(1)   'first preset outline style
    Case cdrEllipseShape
        s.Outline.Style = OutlineStyles(2)   'second preset outline style
    Case cdrPolygonShape
        s.Outline.Style = OutlineStyles(3)   'third preset outline style
    End Select
    End If
Next s
End Sub
```

{button ,AL(`CLS_Outline;FNC_Style')} **Related Topics**

Outline.StartArrow

Property **StartArrow** AS **ArrowHead**

Outline

Description

The **StartArrow** property returns or sets the start arrow of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

Line-ending shapes let you design the start or end point of a curve object. You can apply, edit, or create arrowheads using the Edit Arrowhead dialog box. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes. New line-ending shapes are added to the top of the list of line styles.

Example

The following code example applies an arrowhead with an index of 5 to the beginning of each selected object:

```
Sub ItemArrowHead()
Dim s As Shape
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
    s.Outline.StartArrow = ArrowHeads.Item(5)
    'can also be referenced as s.Outline.StartArrow = ArrowHeads (5)
    End If
Next s
End Sub
```

{button ,AL(`CLS_Outline;FNC_StartArrow')} **Related Topics**

Outline.EndArrow

Property **EndArrow** AS **ArrowHead**

Outline

Description

The **EndArrow** property returns or sets the end arrow of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

Line-ending shapes let you design the start or end point of a curve object. You can apply, edit, or create arrowheads using the Edit Arrowhead dialog box. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes. New line-ending shapes are added to the top of the list of line styles.

Example

The following code example applies an arrowhead with an index of 5 to the end of each selected object:

```
Sub ItemArrowHead()
Dim s As Shape
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
    s.Outline.EndArrow = ArrowHeads.Item(5)
    'can also be referenced as s.Outline.EndArrow = ArrowHeads (5)
    End If
Next s
End Sub
```

{button ,AL(`CLS_Outline;FNC_EndArrow')} **Related Topics**

Outline.NibStretch

Property **NibStretch** AS Long

Outline

Description

The **NibStretch** property returns or sets a numerical value that contributes to the thickness of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

A Nib is the tip of a brush tool. You can set the size, shape, and transparency of a nib. You can give an object a hand-drawn look using Calligraphy settings. By adjusting these settings, you can vary the thickness of an object's outline. Reducing the value makes square nibs rectangular and round nibs oval, creating more pronounced calligraphic effects.

The default value of the **NibStretch** property is 100.

Example

The following code example displays the **NibStretch** value of the outline in the active shape of CorelDRAW:

```
Sub Outline()
MsgBox ActiveShape.Outline.NibStretch
End Sub
```

{button ,AL(`CLS_Outline;FNC_NibStretch')} **Related Topics**

Outline.NibAngle

Property **NibAngle** AS Double

Outline

Description

The **NibAngle** property returns or sets a numerical value that contributes to the angle of an object's outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

A Nib is the tip of a brush tool. You can set the size, shape, and transparency of a nib. You can give an object a hand-drawn look using Calligraphy settings. The angle controls the orientation of the pen in relation to the drawing surface. Reducing the value makes square nibs rectangular and round nibs oval, creating more pronounced calligraphic effects.

The default value of the **NibAngle** property is 0.

Example

The following code example displays the **NibAngle** value of the outline in the active shape of CorelDRAW:

```
Sub Outline()
MsgBox ActiveShape.Outline.NibAngle
End Sub
```

{button ,AL(`CLS_Outline;FNC_NibAngle')} **Related Topics**

Outline.BehindFill

Property **BehindFill** AS Boolean

Outline

Description

The **BehindFill** property returns a True or False value that indicates if an outline's fill is behind the outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

Use the **BehindFill** property to apply an outline to a stylized font such as a script font. Normally, outlines are applied after the fill is applied. Half of the outline lies inside the object, while the other half lies outside the object. When you enable the Behind Fill check box, the outline is drawn first, then the fill is placed on top of the outline. Consequently, half of the outline is covered by the fill.

If the **BehindFill** property is set to True, the fill appears on top of the outline.

Example

The following code example sets the **BehindFill** property of the active shape to True:

```
Sub Outline()
ActiveShape.Outline.BehindFill = True
End Sub
```

{button ,AL(`CLS_Outline;FNC_BehindFill')} **Related Topics**

Outline.LineCaps

Property **LineCaps** AS **cdrOutlineLineCaps**

Outline

Description

The **LineCaps** property returns or sets the endpoint style of an open path outline in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

An open path is a line or curve whose start point and end point are not connected. If you apply a fill to an open path, it is not visible unless the path is closed.

The line cap style affects the appearance of the endpoints of open paths. Setting the line cap shape to Rounded or Extended makes the line slightly longer. The **LineCaps** property returns a value of cdrOutlineLineCaps.

An outline's line cap can be of the following styles:

Square line caps - Enable to set squared-off (truncated) ends that are perpendicular to the path.

Rounded line caps - Enable to set the diameter of the cap equal to the width of the line. This causes the line to be slightly longer.

Extended square line caps - Enable to set square ends that extend half of the line width beyond the end of the line.

Example

The follwing code example sets the line caps type of the active shape's outline to square line caps:

```
Sub Outline()
ActiveShape.Outline.LineCaps = cdrOutlineSquareLineCaps
End Sub
```

{button ,AL(`CLS_Outline;FNC_LineCaps')} **Related Topics**

cdrOutlineButtLineCaps=0

cdrOutlineRoundLineCaps=1

cdrOutlineSquareLineCaps=2

Outline.LineJoin

Property **LineJoin** AS **cdrOutlineLineJoin**

Outline

Description

The **LineJoin** property returns or sets an outline's join type in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

The **LineJoin** property sets the appearance of lines that intersect each other in an outline. When two lines meet at a sharp angle and form a spike that extends beyond the intersection of the lines, the miter limit controls when the application switches from a mitered (pointed) join to a beveled (squared-off) joint.

Any corner that is less than the Miter Limit will have a beveled (squared-off) point. Corner joints above the limit will come to a mitered (sharp) point. This limit prevents corners from extending far beyond the actual corner at small angles, such as when a text character comes to a spike, for example, in the letter "M."

The **LineJoin** property returns a value of cdrOutlineLineJoin.

Example

The follwing code example sets the join type of the active shape's outline to a round line join:

```
Sub Outline()
ActiveShape.Outline.LineJoin = cdrOutlineRoundLineJoin
End Sub
```

{button ,AL(`CLS_Outline;FNC_LineJoin')} **Related Topics**

cdrOutlineMiterLineJoin=0

cdrOutlineRoundLineJoin=1

cdrOutlineBevelLineJoin=2

Outline.ScaleWithShape

Property **ScaleWithShape** AS Boolean

Outline

Description

The **ScaleWithShape** property returns or sets a True or False value that indicates if an object's outline maintains the size proportions of the object in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

Scaling allows you to change an object's horizontal and vertical dimensions or to maintain the aspect ratio. Scaling alters the object's dimensions by a specified percentage.

Example

The following code example scales the active shape's outline to maintain its proportions when the shape is altered in CorelDRAW:

```
Sub Outline()
ActiveShape.Outline.ScaleWithShape = True
End Sub
```

{button ,AL(`CLS_Outline;FNC_ScaleWithShape')} **Related Topics**

Outline.SetProperties

Sub **SetProperties**(ByVal **Width** AS Double, ByRef Style AS **OutlineStyle**, ByRef Color AS **Color**, ByRef StartArrow AS **ArrowHead**, ByRef EndArrow AS **ArrowHead**, ByVal BehindFill AS Boolean, ByVal ScaleWithShape AS Boolean, ByVal LineCaps AS **cdrOutlineLineCaps**, ByVal LineJoin AS **cdrOutlineLineJoin**, ByVal NibAngle AS Double, ByVal NibStretch AS Long)

Outline

Description

The **SetProperties** method allows you to set all the properties of an outline object in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill:

In order to use the **SetProperties** method, you must pass the following parameters:

Parameters	Description
Width	The **Width** property returns or sets a numerical value that indicates the width of an object's outline in CorelDRAW. An outline's width is measured in points.
Style	The **Style** property returns or sets the outline style of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
Color	The **Color** property returns or sets the color of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
StartArrow	The **StartArrow** property returns or sets the start arrow of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
EndArrow	The **EndArrow** property returns or sets the end arrow of an object's outline in CorelDRAW. This value is optional and the default value is Nothing (0).
BehindFill	The **BehindFill** property returns a True or False value that indicates if an outline's fill is behind the outline in CorelDRAW. This value is optional and the default value is False.
ScaleWithShape	The **ScaleWithShape** property returns or sets a True or False value that indicates if an object's outline maintains the size proportions of the object in CorelDRAW. This value is optional and the default value is False.
LineCaps	The **LineCaps** property returns or sets the endpoint style of an open path outline in CorelDRAW. This value is optional. It returns a value of cdrOutlineLineCaps and the default value is cdrOutlineButtLineCaps (0)
LineJoin	The **LineJoin** property sets the appearance of lines that intersect each other in an outline. This value returns cdrOutlineLineJoin. The default value is cdrOutlineMiterLineJoin.
NibAngle	The **NibAngle** property returns or sets a numerical value that contributes to the angle of an object's outline in CorelDRAW. This value is optional and the default value is 0.
NibStretch	The **NibStretch** property returns or sets a numerical value that contributes to the thickness of an object's outline in CorelDRAW. This value is optional and the default value is 100.

{button ,AL(`CLS_Outline;FNC_SetProperties')} **Related Topics**

OutlineStyle properties

OutlineStyle Legend

- DashCount
- DashLength
- GapLength
- Index

OutlineStyle

Class **OutlineStyle**

Properties Referenced by

The **OutlineStyle** class defines the characteristics of outline objects and describes the look and behavior of the objects through its properties and methods. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

CorelDRAW comes with many different line styles. Line styles are preset lines that have different attributes, such as dotted lines, or dashed lines. Applying a line style does not change the shape of the line or the amount of the space it occupies on the drawing page. You can edit an existing line style to meet your needs.

{button ,AL(`CLS_OutlineStyle')} **Related Topics**

An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

OutlineStyle.Index

Property **Index** AS Long

OutlineStyle

Description

The **Index** property returns a value associated with an outline style in CorelDRAW. An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

The index number is a unique reference to a dash and gap pair in an outline style. In an outline, there can be 1 to five pairs of dashes and gaps. Therefore, the **Index** property can have a value of 1 to 5. For example, `.Outline.Style.DashLength(1) = 3` sets the dash length of the first dash in a dash/gap pair to 3, where 3 is the measurement equal to the outline width. The gap portion of the dash/gap pair may appear as `.Outline.Style.GapLength(1) = 2`. If the document unit for the drawing is inches, the width of the outline style is measured in inches as well.

The **Index** property returns a Read-Only value.

Example

The following code example applies a custom outline style to a selected shape in CorelDRAW. The new outline appears as a small dash followed by a long dash, with equal gaps between each dash. The length values are in proportion to the width of the outline style. A value of 1 is equal to one width measurement, in document units:

```
Sub StyleIndex()
With ActiveShape.Outline
    .Width = .03
    .Style.DashCount = 2             'two dashes and gaps appear in each dash/gap pair
    .Style.DashLength(1) = 1  'first dash is .03
    .Style.DashLength(2) = 10 'second dash is .3 (.03*10)
    .Style.GapLength(1) = 4          'first gap is .12 (.03*4)
    .Style.GapLength(2) = 4          'second gap is .12 (.03*4)
End With
End Sub
```

{button ,AL(`CLS_OutlineStyle;FNC_Index')} **Related Topics**

A dash is a broken line segment in an outline style.

A gap is a space between dashes in an outline style.

OutlineStyle.DashCount

Property **DashCount** AS Long

OutlineStyle

Description

The **DashCount** property returns or sets a value associated with the number of dashes and gaps in an outline style in CorelDRAW. An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

The **DashCount** property specifies how many pairs of dashes and gaps appear in an outline style. In an outline style, the number of dash pairs can range from 1 to 5.

For example, `.Outline.Style.DashCount = 3` sets the dash count to 3. This means that the new outline is made up of a series of 3 dash styles and 3 gap styles. The style of each dash and gap may be customized with the **DashLength** property and the **GapLength** property.

Example

The following code example applies a custom outline style to a selected shape in CorelDRAW. The new outline appears as a small dash followed by a long dash, with equal gaps between each dash. The length values are in proportion to the width of the outline style. A value of 1 is equal to one width measurement, in document units:

```
Sub StyleIndex()
With ActiveShape.Outline
    .Width = .03
    .Style.DashCount = 2             'two dashes appear in each dash pair
    .Style.DashLength(1) = 1  'first dash is .03
    .Style.DashLength(2) = 10 'second dash is .3 (.03*10)
    .Style.GapLength(1) = 4          'first gap is .12 (.03*4)
    .Style.GapLength(2) = 4          'second gap is .12 (.03*4)
End With
End Sub
```

{button ,AL(`CLS_OutlineStyle;FNC_DashCount')} **Related Topics**

OutlineStyle.DashLength

Property **DashLength**(ByVal **Index** AS Long) AS Long

OutlineStyle

Description

The **DashLength** property returns or sets a value associated with the length of dashes in an outline style in CorelDRAW. An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

The **DashLength** property specifies the length of each dash in an outline style. In an outline style, the number of dash/gap pairs can range from 1 to 5. The length of each dash is measured in proportion to the width of the outline style. All measurements are in document units.

For example, `.Outline.Style.DashLength(1) = 2` sets the length of the first dash in a dash/gap pair to 2. If the width of an outline is .5 (document unit is inches), the actual length of the first dash in the dash/gap pair is 1 inch (.5*2).

In order to set the length of a dash with **DashLength** property, you must past the **Index** as a parameter:

Parameters	Description
Index	**Index** is a unique reference to a dash or gap pair in an outline style. In an outline, there can be 1 to five pairs of dashes or gaps. Therefore, the **Index** property can have a value of 1 to 5. For example, `.Outline.Style.DashLength(1) = 3` sets the dash length of the first dash in a dash pair to 3, where 3 is the measurement equal to the outline width.

Example

The following code example applies a custom outline style to a selected shape in CorelDRAW. The new outline appears as a small dash followed by a long dash, with equal gaps between each dash. The length values are in proportion to the width of the outline style. A value of 1 is equal to one width measurement, in document units:

```
Sub StyleIndex()
With ActiveShape.Outline
    .Width = .03
    .Style.DashCount = 2             'two dashes appear in each dash pair
    .Style.DashLength(1) = 1  'first dash is .03
    .Style.DashLength(2) = 10 'second dash is .3 (.03*10)
    .Style.GapLength(1) = 4          'first gap is .12 (.03*4)
    .Style.GapLength(2) = 4          'second gap is .12 (.03*4)
End With
End Sub
```

{button ,AL(`CLS_OutlineStyle;FNC_DashLength')} **Related Topics**

OutlineStyle.GapLength

Property **GapLength**(ByVal **Index** AS Long) AS Long

OutlineStyle

Description

The **GapLength** property returns or sets a value associated with the length of a gap in an outline style in CorelDRAW. An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

The **GapLength** property specifies the length of each gap in an outline style. In an outline style, the number of dash/gap pairs can range from 1 to 5. The length of each gap is measured in proportion to the width of the outline style. All measurements are in document units.

For example, `.Outline.Style.GapLength(1) = 2` sets the length of the first gap in a dash/gap pair to 2. If the width of an outline is .5 (document unit is inches), the actual length of the first gap in the dash/gap pair is 1 inch (.5*2).

In order to set the length of a gap with **GapLength** property, you must past the **Index** as a parameter:

Parameters	Description
Index	**Index** is a unique reference to a dash or gap pair in an outline style. In an outline, there can be 1 to five pairs of dashes or gaps. Therefore, the **Index** property can have a value of 1 to 5. For example, `.Outline.Style.GapLength(1) = 3` sets the gap length of the first gap in a dash pair to 3, where 3 is the measurement equal to the outline width.

Example

The following code example applies a custom outline style to a selected shape in CorelDRAW. The new outline appears as a small dash followed by a long dash, with equal gaps between each dash. The length values are in proportion to the width of the outline style. A value of 1 is equal to one width measurement, in document units:

```
Sub StyleIndex()
With ActiveShape.Outline
    .Width = .03
    .Style.DashCount = 2             'two dashes appear in each dash pair
    .Style.DashLength(1) = 1  'first dash is .03
    .Style.DashLength(2) = 10 'second dash is .3 (.03*10)
    .Style.GapLength(1) = 4          'first gap is .12 (.03*4)
    .Style.GapLength(2) = 4          'second gap is .12 (.03*4)
End With
End Sub
```

{button ,AL(`CLS_OutlineStyle;FNC_GapLength')} **Related Topics**

Printer properties

Printer Legend

- ColorEnabled
- Default
- Description
- Name
- Port
- PostScriptEnabled
- Ready
- Type

Printer methods

Printer Legend

ShowDialog

Printer

Class **Printer**

Properties Methods Referenced By

Printer Class

{button ,AL(`CLS_Printer')} **Related Topics**

Printer.ColorEnabled

Property **ColorEnabled** As Boolean

property ColorEnabled

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_ColorEnabled')} **Related Topics**

Printer.Default

Property **Default** As Boolean

property Default

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Default')} **Related Topics**

Printer.Description

Property **Description** As String

property Description

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Description')} **Related Topics**

Printer.Name

Property **Name** As String

property Name

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Name')} **Related Topics**

Printer.Port

Property **Port** As String

property Port

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Port')} **Related Topics**

Printer.PostScriptEnabled

Property **PostScriptEnabled** As Boolean

property PostScriptEnabled

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_PostScriptEnabled')} **Related Topics**

Printer.Ready

Property **Ready** As Boolean

property Ready

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Ready')} **Related Topics**

Printer.Type

Property **Type** As String

property Type

Member of **Printer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_Type')} **Related Topics**

Printer.ShowDialog

Sub **ShowDialog**()

method ShowDialog

Member of **Printer**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printer;FNC_ShowDialog')} **Related Topics**

Printers properties

Printers Legend

- Count

- Default

-
- Item

Printers

Class **Printers**

Properties Referenced By

Printers Class

{button ,AL(`CLS_Printers')} **Related Topics**

Printers.Count

Property **Count** As Long

property Count

Member of **Printers**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printers;FNC_Count')} **Related Topics**

Printers.Default

Property **Default** As **Printer**

property Default

Member of **Printers**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printers;FNC_Default')} **Related Topics**

Printers.Item

Property **Item**(ByVal **nIndex** As Long) As **Printer**

property Item

Default member of **Printers**

Read-Only

Parameters	Description
nIndex	Description of nIndex goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Printers;FNC_Item')} **Related Topics**

PrintJob properties

PrintJob Legend

- Documents
- Pages
- Settings

PrintJob methods

PrintJob Legend

- AddDocument
- Clear
- PrintOut

PrintJob

Class **PrintJob**

Properties Methods Referenced By

PrintJob Class

{button ,AL(`CLS_PrintJob')} **Related Topics**

PrintJob.Documents

Property **Documents** As **PrintDocuments**

property Documents

Member of **PrintJob**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_Documents')} **Related Topics**

PrintJob.Pages

Property **Pages** As **PrintPages**

property Pages

Member of **PrintJob**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_Pages')} **Related Topics**

PrintJob.Settings

Property **Settings** As **PrintSettings**

property Settings

Member of **PrintJob**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_Settings')} **Related Topics**

PrintJob.AddDocument

Sub **AddDocument**(ByVal **Document** As **Document**, [ByVal PageRange As String])

method AddDocument

Member of **PrintJob**

Parameters	Description
Document	Description of Document goes here (**in**)
PageRange	Description of PageRange goes here (**in**) Optional

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_AddDocument')} **Related Topics**

PrintJob.Clear

Sub **Clear**()

method Clear

Member of **PrintJob**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_Clear')} **Related Topics**

PrintJob.PrintOut

Sub **PrintOut**()

method PrintOut

Member of **PrintJob**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintJob;FNC_PrintOut')} **Related Topics**

PrintOptions properties

PrintOptions Legend

- AppInfo
- BitmapColorMode
- ColorMode
- ColorResolution
- DownsampleColor
- DownsampleGray
- DownsampleMono
- DriverInfo
- FontInfo
- FountainSteps
- GrayResolution
- JobInformation
- LinkInfo
- MarksToPage
- MonoResolution
- PrintBitmaps
- PrintJobInfo
- PrintText
- PrintVectors
- RasterizePage
- RasterizeResolution
- SepsInfo
- TextInBlack
- UseColorProfile

PrintOptions

Class **PrintOptions**

Properties Referenced By

PrintOptions Class

{button ,AL(`CLS_PrintOptions')} **Related Topics**

PrintOptions.AppInfo

Property **AppInfo** As Boolean

property AppInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_AppInfo')} **Related Topics**

PrintOptions.BitmapColorMode

Property **BitmapColorMode** As **PrnBitmapColorMode**

property BitmapColorMode

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_BitmapColorMode')} **Related Topics**

PrintOptions.ColorMode

Property **ColorMode** As **PrnColorMode**

property ColorMode

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_ColorMode')} **Related Topics**

PrintOptions.ColorResolution

Property **ColorResolution** As Long

property ColorResolution

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_ColorResolution')} **Related Topics**

PrintOptions.DownsampleColor

Property **DownsampleColor** As Boolean

property DownsampleColor

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_DownsampleColor')} **Related Topics**

PrintOptions.DownsampleGray

Property **DownsampleGray** As Boolean

property DownsampleGray

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_DownsampleGray')} **Related Topics**

PrintOptions.DownsampleMono

Property **DownsampleMono** As Boolean

property DownsampleMono

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_DownsampleMono')} **Related Topics**

PrintOptions.DriverInfo

Property **DriverInfo** As Boolean

property DriverInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_DriverInfo')} **Related Topics**

PrintOptions.FontInfo

Property **FontInfo** As Boolean

property FontInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_FontInfo')} **Related Topics**

PrintOptions.FountainSteps

Property **FountainSteps** As Long

property FountainSteps

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_FountainSteps')} **Related Topics**

PrintOptions.GrayResolution

Property **GrayResolution** As Long

property GrayResolution

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_GrayResolution')} **Related Topics**

PrintOptions.JobInformation

Property **JobInformation** As Boolean

property JobInformation

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_JobInformation')} **Related Topics**

PrintOptions.LinkInfo

Property **LinkInfo** As Boolean

property LinkInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_LinkInfo')} **Related Topics**

PrintOptions.MarksToPage

Property **MarksToPage** As Boolean

property MarksToPage

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_MarksToPage')} **Related Topics**

PrintOptions.MonoResolution

Property **MonoResolution** As Long

property MonoResolution

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_MonoResolution')} **Related Topics**

PrintOptions.PrintBitmaps

Property **PrintBitmaps** As Boolean

property PrintBitmaps

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_PrintBitmaps')} **Related Topics**

PrintOptions.PrintJobInfo

Property **PrintJobInfo** As Boolean

property PrintJobInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_PrintJobInfo')} **Related Topics**

PrintOptions.PrintText

Property **PrintText** As Boolean

property PrintText

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_PrintText')} **Related Topics**

PrintOptions.PrintVectors

Property **PrintVectors** As Boolean

property PrintVectors

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_PrintVectors')} **Related Topics**

PrintOptions.RasterizePage

Property **RasterizePage** As Boolean

property RasterizePage

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_RasterizePage')} **Related Topics**

PrintOptions.RasterizeResolution

Property **RasterizeResolution** As Long

property RasterizeResolution

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_RasterizeResolution')} **Related Topics**

PrintOptions.SepsInfo

Property **SepsInfo** As Boolean

property SepsInfo

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_SepsInfo')} **Related Topics**

PrintOptions.TextInBlack

Property **TextInBlack** As Boolean

property TextInBlack

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_TextInBlack')} **Related Topics**

PrintOptions.UseColorProfile

Property **UseColorProfile** As Boolean

property UseColorProfile

Member of **PrintOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintOptions;FNC_UseColorProfile')} **Related Topics**

PrintPages properties

PrintPages Legend

- Count

PrintPages

Class **PrintPages**

Properties Referenced By

PrintPages Class

{button ,AL(`CLS_PrintPages')} **Related Topics**

PrintPages.Count

Property **Count** As Long

property Count

Member of **PrintPages**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPages;FNC_Count')} **Related Topics**

PrintPostScript properties

PrintPostScript Legend

- AutoIncreaseFlatness
- AutoIncreaseFountainSteps
- ConformToDSC
- DownloadType1
- Flatness
- JPEGCompression
- JPEGQuality
- Level
- MaintainOPILinks
- MaxPointsPerCurve
- OptimizeFountainFills
- PDFBookmarks
- PDFHyperlinks
- PDFStartup
- ResolveDCSLinks
- ScreenFrequency
- TrueTypeToType1

PrintPostScript

Class **PrintPostScript**

Properties Referenced By

PrintPostScript Class

{button ,AL(`CLS_PrintPostScript')} **Related Topics**

PrintPostScript.AutoIncreaseFlatness

Property **AutoIncreaseFlatness** As Boolean

property AutoIncreaseFlatness

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_AutoIncreaseFlatness')} **Related Topics**

PrintPostScript.AutoIncreaseFountainSteps

Property **AutoIncreaseFountainSteps** As Boolean

property AutoIncreaseFountainSteps

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_AutoIncreaseFountainSteps')} **Related Topics**

PrintPostScript.ConformToDSC

Property **ConformToDSC** As Boolean

property ConformToDSC

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_ConformToDSC')} **Related Topics**

PrintPostScript.DownloadType1

Property **DownloadType1** As Boolean

property DownloadType1

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_DownloadType1')} **Related Topics**

PrintPostScript.Flatness

Property **Flatness** As Long

property Flatness

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_Flatness')} **Related Topics**

PrintPostScript.JPEGCompression

Property **JPEGCompression** As Boolean

property JPEGCompression

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_JPEGCompression')} **Related Topics**

PrintPostScript.JPEGQuality

Property **JPEGQuality** As Long

property JPEGQuality

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_JPEGQuality')} **Related Topics**

PrintPostScript.Level

Property **Level** As **PrnPostScriptLevel**

property Level

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_Level')} **Related Topics**

PrintPostScript.MaintainOPILinks

Property **MaintainOPILinks** As Boolean

property MaintainOPILinks

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_MaintainOPILinks')} **Related Topics**

PrintPostScript.MaxPointsPerCurve

Property **MaxPointsPerCurve** As Long

property MaxPointsPerCurve

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_MaxPointsPerCurve')} **Related Topics**

PrintPostScript.OptimizeFountainFills

Property **OptimizeFountainFills** As Boolean

property OptimizeFountainFills

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_OptimizeFountainFills')} **Related Topics**

PrintPostScript.PDFBookmarks

Property **PDFBookmarks** As Boolean

property PDFBookmarks

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_PDFBookmarks')} **Related Topics**

PrintPostScript.PDFHyperlinks

Property **PDFHyperlinks** As Boolean

property PDFHyperlinks

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_PDFHyperlinks')} **Related Topics**

PrintPostScript.PDFStartup

Property **PDFStartup** As **PrnPDFStartup**

property PDFStartup

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_PDFStartup')} **Related Topics**

PrintPostScript.ResolveDCSLinks

Property **ResolveDCSLinks** As Boolean

property ResolveDCSLinks

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_ResolveDCSLinks')} **Related Topics**

PrintPostScript.ScreenFrequency

Property **ScreenFrequency** As Long

property ScreenFrequency

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_ScreenFrequency')} **Related Topics**

PrintPostScript.TrueTypeToType1

Property **TrueTypeToType1** As Boolean

property TrueTypeToType1

Member of **PrintPostScript**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPostScript;FNC_TrueTypeToType1')} **Related Topics**

PrintPrepress properties

PrintPrepress Legend

- ColorCalibrationBar
- CropMarks
- Densities
- DensitometerScale
- ExteriorCropMarks
- FileInfo
- InfoWithinPage
- Invert
- JobName
- Mirror
- PageNumbers
- RegistrationMarks
- RegistrationStyle

PrintPrepress

Class **PrintPrepress**

Properties Referenced By

PrintPrepress Class

{button ,AL(`CLS_PrintPrepress')} **Related Topics**

PrintPrepress.ColorCalibrationBar

Property **ColorCalibrationBar** As Boolean

property ColorCalibrationBar

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_ColorCalibrationBar')} **Related Topics**

PrintPrepress.CropMarks

Property **CropMarks** As Boolean

property CropMarks

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_CropMarks')} **Related Topics**

PrintPrepress.Densities

Property **Densities**(ByVal **Index** As Long) As Long

property Densities

Member of **PrintPrepress**

Parameters	Description
Index	Description of Index goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_Densities')} **Related Topics**

PrintPrepress.DensitometerScale

Property **DensitometerScale** As Boolean

property DensitometerScale

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_DensitometerScale')} **Related Topics**

PrintPrepress.ExteriorCropMarks

Property **ExteriorCropMarks** As Boolean

property ExteriorCropMarks

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_ExteriorCropMarks')} **Related Topics**

PrintPrepress.FileInfo

Property **FileInfo** As Boolean

property FileInfo

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_FileInfo')} **Related Topics**

PrintPrepress.InfoWithinPage

Property **InfoWithinPage** As Boolean

property InfoWithinPage

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_InfoWithinPage')} **Related Topics**

PrintPrepress.Invert

Property **Invert** As Boolean

property Invert

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_Invert')} **Related Topics**

PrintPrepress.JobName

Property **JobName** As String

property JobName

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_JobName')} **Related Topics**

PrintPrepress.Mirror

Property **Mirror** As Boolean

property Mirror

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_Mirror')} **Related Topics**

PrintPrepress.PageNumbers

Property **PageNumbers** As Boolean

property PageNumbers

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_PageNumbers')} **Related Topics**

PrintPrepress.RegistrationMarks

Property **RegistrationMarks** As Boolean

property RegistrationMarks

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_RegistrationMarks')} **Related Topics**

PrintPrepress.RegistrationStyle

Property **RegistrationStyle** As **PrnRegistrationStyle**

property RegistrationStyle

Member of **PrintPrepress**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintPrepress;FNC_RegistrationStyle')} **Related Topics**

PrintSeparations properties

PrintSeparations Legend

- AdvancedSettings
- AlwaysOverprintBlack
- AutoSpreadAmount
- AutoSpreadFixed
- AutoSpreading
- AutoSpreadTextAbove
- BasicScreen
- EmptyPlates
- Enabled
- HalftoneType
- Hexachrome
- InColor
- Plates
- PreserveOverprints
- Resolution
- ScreenTechnology
- SpotToCMYK

PrintSeparations

Class **PrintSeparations**

Properties Referenced By

PrintSeparations Class

{button ,AL(`CLS_PrintSeparations')} **Related Topics**

PrintSeparations.AdvancedSettings

Property **AdvancedSettings** As Boolean

property AdvancedSettings

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AdvancedSettings')} **Related Topics**

PrintSeparations.AlwaysOverprintBlack

Property **AlwaysOverprintBlack** As Boolean

property AlwaysOverprintBlack

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AlwaysOverprintBlack')} **Related Topics**

PrintSeparations.AutoSpreadAmount

Property **AutoSpreadAmount** As Double

property AutoSpreadAmount

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AutoSpreadAmount')} **Related Topics**

PrintSeparations.AutoSpreadFixed

Property **AutoSpreadFixed** As Boolean

property AutoSpreadFixed

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AutoSpreadFixed')} **Related Topics**

PrintSeparations.AutoSpreading

Property **AutoSpreading** As Boolean

property AutoSpreading

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AutoSpreading')} **Related Topics**

PrintSeparations.AutoSpreadTextAbove

Property **AutoSpreadTextAbove** As Double

property AutoSpreadTextAbove

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_AutoSpreadTextAbove')} **Related Topics**

PrintSeparations.BasicScreen

Property **BasicScreen** As String

property BasicScreen

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_BasicScreen')} **Related Topics**

PrintSeparations.EmptyPlates

Property **EmptyPlates** As Boolean

property EmptyPlates

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_EmptyPlates')} **Related Topics**

PrintSeparations.Enabled

Property **Enabled** As Boolean

property Enabled

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_Enabled')} **Related Topics**

PrintSeparations.HalftoneType

Property **HalftoneType** As String

property HalftoneType

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_HalftoneType')} **Related Topics**

PrintSeparations.Hexachrome

Property **Hexachrome** As Boolean

property Hexachrome

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_Hexachrome')} **Related Topics**

PrintSeparations.InColor

Property **InColor** As Boolean

property InColor

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_InColor')} **Related Topics**

PrintSeparations.Plates

Property **Plates** As **SeparationPlates**

property Plates

Member of **PrintSeparations**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_Plates')} **Related Topics**

PrintSeparations.PreserveOverprints

Property **PreserveOverprints** As Boolean

property PreserveOverprints

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_PreserveOverprints')} **Related Topics**

PrintSeparations.Resolution

Property **Resolution** As Long

property Resolution

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_Resolution')} **Related Topics**

PrintSeparations.ScreenTechnology

Property **ScreenTechnology** As String

property ScreenTechnology

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_ScreenTechnology')} **Related Topics**

PrintSeparations.SpotToCMYK

Property **SpotToCMYK** As Boolean

property SpotToCMYK

Member of **PrintSeparations**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSeparations;FNC_SpotToCMYK')} **Related Topics**

PrintSettings properties

PrintSettings Legend

- Collate
- Copies
- FileMode
- FileName
- ForMac
- Options
- PageRange

- PostScript
  - PPDFile

- Prepress
  - Printer
  - PrintRange
  - PrintToFile

- Separations

- Trapping
  - UsePPD

PrintSettings methods

PrintSettings Legend

- Load
- Reset
- Save
- ShowDialog

PrintSettings

Class **PrintSettings**

Properties Methods Referenced By

PrintSettings Class

{button ,AL(`CLS_PrintSettings')} **Related Topics**

PrintSettings.Collate

Property **Collate** As Boolean

property Collate

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Collate')} **Related Topics**

PrintSettings.Copies

Property **Copies** As Long

property Copies

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Copies')} **Related Topics**

PrintSettings.FileMode

Property **FileMode** As **PrnFileMode**

property FileMode

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_FileMode')} **Related Topics**

PrintSettings.FileName

Property **FileName** As String

property FileName

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_FileName')} **Related Topics**

PrintSettings.ForMac

Property **ForMac** As Boolean

property ForMac

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_ForMac')} **Related Topics**

PrintSettings.Options

Property **Options** As **PrintOptions**

property Options

Member of **PrintSettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Options')} **Related Topics**

PrintSettings.PageRange

Property **PageRange** As String

property PageRange

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_PageRange')} **Related Topics**

PrintSettings.PostScript

Property **PostScript** As **PrintPostScript**

property PostScript

Member of **PrintSettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_PostScript')} **Related Topics**

PrintSettings.PPDFile

Property **PPDFile** As String

property PPDFile

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_PPDFile')} **Related Topics**

PrintSettings.Prepress

Property **Prepress** As **PrintPrepress**

property Prepress

Member of **PrintSettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Prepress')} **Related Topics**

PrintSettings.Printer

Property **Printer** As **Printer**

property Printer

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Printer')} **Related Topics**

PrintSettings.PrintRange

Property **PrintRange** As **PrnPrintRange**

property PrintRange

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_PrintRange')} **Related Topics**

PrintSettings.PrintToFile

Property **PrintToFile** As Boolean

property PrintToFile

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_PrintToFile')} **Related Topics**

PrintSettings.Separations

Property **Separations** As **PrintSeparations**

property Separations

Member of **PrintSettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Separations')} **Related Topics**

PrintSettings.Trapping

Property **Trapping** As **PrintTrapping**

property Trapping

Member of **PrintSettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Trapping')} **Related Topics**

PrintSettings.UsePPD

Property **UsePPD** As Boolean

property UsePPD

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_UsePPD')} **Related Topics**

PrintSettings.Load

Function **Load**(ByVal **FileName** As String) As Boolean

method Load

Member of **PrintSettings**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Load')} **Related Topics**

PrintSettings.Reset

Sub **Reset**()

method Reset

Member of **PrintSettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Reset')} **Related Topics**

PrintSettings.Save

Function **Save**(ByVal **FileName** As String) As Boolean

method Save

Member of **PrintSettings**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_Save')} **Related Topics**

PrintSettings.ShowDialog

Function **ShowDialog**() As Boolean

method ShowDialog

Member of **PrintSettings**

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintSettings;FNC_ShowDialog')} **Related Topics**

PrintTrapping properties

PrintTrapping Legend

- BlackColorLimit
- BlackDensityLimit
- BlackWidth
- ColorScaling
- Enabled
- ImageTrap
- InternalImageTrapping
- Layers
- MinResolution
- ObjectsToImage
- SlidingTrapLimit
- StepLimit
- Width

PrintTrapping

Class **PrintTrapping**

Properties Referenced By

PrintTrapping Class

{button ,AL(`CLS_PrintTrapping')} **Related Topics**

PrintTrapping.BlackColorLimit

Property **BlackColorLimit** As Long

property BlackColorLimit

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_BlackColorLimit')} **Related Topics**

PrintTrapping.BlackDensityLimit

Property **BlackDensityLimit** As Double

property BlackDensityLimit

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_BlackDensityLimit')} **Related Topics**

PrintTrapping.BlackWidth

Property **BlackWidth** As Double

property BlackWidth

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_BlackWidth')} **Related Topics**

PrintTrapping.ColorScaling

Property **ColorScaling** As Long

property ColorScaling

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_ColorScaling')} **Related Topics**

PrintTrapping.Enabled

Property **Enabled** As Boolean

property Enabled

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_Enabled')} **Related Topics**

PrintTrapping.ImageTrap

Property **ImageTrap** As **PrnImageTrap**

property ImageTrap

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_ImageTrap')} **Related Topics**

PrintTrapping.InternalImageTrapping

Property **InternalImageTrapping** As Boolean

property InternalImageTrapping

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_InternalImageTrapping')} **Related Topics**

PrintTrapping.Layers

Property **Layers** As **TrapLayers**

property Layers

Member of **PrintTrapping**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_Layers')} **Related Topics**

PrintTrapping.MinResolution

Property **MinResolution** As Long

property MinResolution

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_MinResolution')} **Related Topics**

PrintTrapping.ObjectsToImage

Property **ObjectsToImage** As Boolean

property ObjectsToImage

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_ObjectsToImage')} **Related Topics**

PrintTrapping.SlidingTrapLimit

Property **SlidingTrapLimit** As Long

property SlidingTrapLimit

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_SlidingTrapLimit')} **Related Topics**

PrintTrapping.StepLimit

Property **StepLimit** As Long

property StepLimit

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_StepLimit')} **Related Topics**

PrintTrapping.Width

Property **Width** As Double

property Width

Member of **PrintTrapping**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintTrapping;FNC_Width')} **Related Topics**

SeparationPlate properties

SeparationPlate Legend

- Angle
- Color
- Enabled
- Frequency
- OverprintGraphic
- OverprintText

- Type

SeparationPlate

Class **SeparationPlate**

Properties Referenced By

SeparationPlate Class

{button ,AL(`CLS_SeparationPlate')} **Related Topics**

SeparationPlate.Angle

Property **Angle** As Double

property Angle

Member of **SeparationPlate**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_Angle')} **Related Topics**

SeparationPlate.Color

Property **Color** As String

property Color

Member of **SeparationPlate**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_Color')} **Related Topics**

SeparationPlate.Enabled

Property **Enabled** As Boolean

property Enabled

Member of **SeparationPlate**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_Enabled')} **Related Topics**

SeparationPlate.Frequency

Property **Frequency** As Double

property Frequency

Member of **SeparationPlate**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_Frequency')} **Related Topics**

SeparationPlate.OverprintGraphic

Property **OverprintGraphic** As Boolean

property OverprintGraphic

Member of **SeparationPlate**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_OverprintGraphic')} **Related Topics**

SeparationPlate.OverprintText

Property **OverprintText** As Boolean

property OverprintText

Member of **SeparationPlate**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_OverprintText')} **Related Topics**

SeparationPlate.Type

Property **Type** As **PrnPlateType**

property Type

Member of **SeparationPlate**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlate;FNC_Type')} **Related Topics**

SeparationPlates properties

SeparationPlates Legend

- Count

- Item

SeparationPlates

Class **SeparationPlates**

Properties Referenced By

SeparationPlates Class

{button ,AL(`CLS_SeparationPlates')} **Related Topics**

SeparationPlates.Count

Property **Count** As Long

property Count

Member of **SeparationPlates**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlates;FNC_Count')} **Related Topics**

SeparationPlates.Item

Property **Item**(ByVal **Index** As Long) As **SeparationPlate**

property Item

Default member of **SeparationPlates**

Read-Only

Parameters	Description
Index	Description of Index goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_SeparationPlates;FNC_Item')} **Related Topics**

PrintDocuments properties

PrintDocuments Legend

- Count

PrintDocuments

Class **PrintDocuments**

Properties Referenced By

PrintDocuments Class

{button ,AL(`CLS_PrintDocuments')} **Related Topics**

PrintDocuments.Count

Property **Count** As Long

property Count

Member of **PrintDocuments**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PrintDocuments;FNC_Count')} **Related Topics**

PDFVBASettings properties

PDFVBASettings Legend

- Author
- BitmapCompression
- Bleed
- Bookmarks
- ColorMode
- ColorProfile
- ColorResolution
- ComplexFillsAsBitmaps
- CompressText
- CropMarks
- DensitometerScales
- DownsampleColor
- DownsampleGray
- DownsampleMono
- EmbedBaseFonts
- EmbedFonts
- EPSAs
- FileInformation
- FountainSteps
- GetEmbedFile

- GetEmbedFilename

- GetEncoding
- GrayResolution
- Halftones
- Hyperlinks
- IncludeBleed
- JPEGQualityFactor
- Keywords
- Linearize
- MaintainOPILinks
- MonoResolution
- Overprints
- PageRange
- pdfVersion
- PublishRange
- RegistrationMarks
- SetEmbedFile
- SetEmbedFilename
- SetEncoding
- SpotColors
- Startup
- Subject
- SubsetFonts
- SubsetPct
- TextAsCurves
- Thumbnails
- TrueTypeToType1
- UseColorProfile

PDFVBASettings methods

PDFVBASettings Legend

- Load
- Reset
- Save
- ShowDialog

PDFVBASettings

Class **PDFVBASettings**

Properties Methods Referenced By

{button ,AL(`CLS_PDFVBASettings')} **Related Topics**

PDFVBASettings.Author

Property **Author** As String

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Author')} **Related Topics**

PDFVBASettings.BitmapCompression

Property **BitmapCompression** As **pdfBitmapCompressionType**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_BitmapCompression')} **Related Topics**

PDFVBASettings.Bleed

Property **Bleed** As Integer

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Bleed')} **Related Topics**

PDFVBASettings.Bookmarks

Property **Bookmarks** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Bookmarks')} **Related Topics**

PDFVBASettings.ColorMode

Property **ColorMode** As **pdfColorMode**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_ColorMode')} **Related Topics**

PDFVBASettings.ColorProfile

Property **ColorProfile** As **pdfColorProfile**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_ColorProfile')} **Related Topics**

PDFVBASettings.ColorResolution

Property **ColorResolution** As Long

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_ColorResolution')} **Related Topics**

PDFVBASettings.ComplexFillsAsBitmaps

Property **ComplexFillsAsBitmaps** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_ComplexFillsAsBitmaps')} **Related Topics**

PDFVBASettings.CompressText

Property **CompressText** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_CompressText')} **Related Topics**

PDFVBASettings.CropMarks

Property **CropMarks** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_CropMarks')} **Related Topics**

PDFVBASettings.DensitometerScales

Property **DensitometerScales** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_DensitometerScales')} **Related Topics**

PDFVBASettings.DownsampleColor

Property **DownsampleColor** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_DownsampleColor')} **Related Topics**

PDFVBASettings.DownsampleGray

Property **DownsampleGray** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_DownsampleGray')} **Related Topics**

PDFVBASettings.DownsampleMono

Property **DownsampleMono** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_DownsampleMono')} **Related Topics**

PDFVBASettings.EmbedBaseFonts

Property **EmbedBaseFonts** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_EmbedBaseFonts')} **Related Topics**

PDFVBASettings.EmbedFonts

Property **EmbedFonts** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_EmbedFonts')} **Related Topics**

PDFVBASettings.EPSAs

Property **EPSAs** As **pdfEPSAs**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_EPSAs')} **Related Topics**

PDFVBASettings.FileInformation

Property **FileInformation** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_FileInformation')} **Related Topics**

PDFVBASettings.FountainSteps

Property **FountainSteps** As Long

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_FountainSteps')} **Related Topics**

PDFVBASettings.GetEmbedFile

Property **GetEmbedFile** As Boolean

Member of **PDFVBASettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_GetEmbedFile')} **Related Topics**

PDFVBASettings.GetEmbedFilename

Property **GetEmbedFilename** As String

Member of **PDFVBASettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_GetEmbedFilename')} **Related Topics**

PDFVBASettings.GetEncoding

Property **GetEncoding** As **pdfEncodingType**

Member of **PDFVBASettings**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_GetEncoding')} **Related Topics**

PDFVBASettings.GrayResolution

Property **GrayResolution** As Long

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_GrayResolution')} **Related Topics**

PDFVBASettings.Halftones

Property **Halftones** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Halftones')} **Related Topics**

PDFVBASettings.Hyperlinks

Property **Hyperlinks** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Hyperlinks')} **Related Topics**

PDFVBASettings.IncludeBleed

Property **IncludeBleed** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_IncludeBleed')} **Related Topics**

PDFVBASettings.JPEGQualityFactor

Property **JPEGQualityFactor** As Integer

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_JPEGQualityFactor')} **Related Topics**

PDFVBASettings.Keywords

Property **Keywords** As String

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Keywords')} **Related Topics**

PDFVBASettings.Linearize

Property **Linearize** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Linearize')} **Related Topics**

PDFVBASettings.MaintainOPILinks

Property **MaintainOPILinks** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_MaintainOPILinks')} **Related Topics**

PDFVBASettings.MonoResolution

Property **MonoResolution** As Long

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_MonoResolution')} **Related Topics**

PDFVBASettings.Overprints

Property **Overprints** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Overprints')} **Related Topics**

PDFVBASettings.PageRange

Property **PageRange** As String

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_PageRange')} **Related Topics**

PDFVBASettings.pdfVersion

Property **pdfVersion** As **pdfVersion**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_pdfVersion')} **Related Topics**

PDFVBASettings.PublishRange

Property **PublishRange** As **pdfExportRange**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_PublishRange')} **Related Topics**

PDFVBASettings.RegistrationMarks

Property **RegistrationMarks** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_RegistrationMarks')} **Related Topics**

PDFVBASettings.SetEmbedFile

Property **SetEmbedFile**(ByVal Boolean)

Member of **PDFVBASettings**

Write-Only

Parameters	Description
	Description of goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SetEmbedFile')} **Related Topics**

PDFVBASettings.SetEmbedFilename

Property **SetEmbedFilename**(ByVal String)

Member of **PDFVBASettings**

Write-Only

Parameters	Description
	Description of goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SetEmbedFilename')} **Related Topics**

PDFVBASettings.SetEncoding

Property **SetEncoding**(ByVal **pdfEncodingType**)

Member of **PDFVBASettings**

Write-Only

Parameters	Description
	Description of goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SetEncoding')} **Related Topics**

PDFVBASettings.SpotColors

Property **SpotColors** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SpotColors')} **Related Topics**

PDFVBASettings.Startup

Property **Startup** As **pdfDisplayOnStart**

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Startup')} **Related Topics**

PDFVBASettings.Subject

Property **Subject** As String

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Subject')} **Related Topics**

PDFVBASettings.SubsetFonts

Property **SubsetFonts** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SubsetFonts')} **Related Topics**

PDFVBASettings.SubsetPct

Property **SubsetPct** As Long

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_SubsetPct')} **Related Topics**

PDFVBASettings.TextAsCurves

Property **TextAsCurves** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_TextAsCurves')} **Related Topics**

PDFVBASettings.Thumbnails

Property **Thumbnails** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Thumbnails')} **Related Topics**

PDFVBASettings.TrueTypeToType1

Property **TrueTypeToType1** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_TrueTypeToType1')} **Related Topics**

PDFVBASettings.UseColorProfile

Property **UseColorProfile** As Boolean

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_UseColorProfile')} **Related Topics**

PDFVBASettings.Load

Function **Load**(ByVal **SettingName** As String) As Boolean

Member of **PDFVBASettings**

Parameters	Description
SettingName	Description of SettingName goes here (**in**)

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Load')} **Related Topics**

PDFVBASettings.Reset

Sub **Reset**()

Member of **PDFVBASettings**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Reset')} **Related Topics**

PDFVBASettings.Save

Function **Save**(ByVal **SettingName** As String) As Boolean

Member of **PDFVBASettings**

Parameters	Description
SettingName	Description of SettingName goes here (**in**)

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_Save')} **Related Topics**

PDFVBASettings.ShowDialog

Function **ShowDialog**() As Boolean

Member of **PDFVBASettings**

Return value

Returns Boolean

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_PDFVBASettings;FNC_ShowDialog')} **Related Topics**

OutlineStyles properties

OutlineStyles Legend

- Application

- Count

-
- Item

- Parent

OutlineStyles methods

OutlineStyles Legend

- Add
- AddStyle
- Remove
- Save

OutlineStyles.Save

Sub **Save**()

Saves changes made to outline styles

Member of **OutlineStyles**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_OutlineStyles;FNC_Save')} **Related Topics**

OutlineStyles.AddStyle

Function **AddStyle**(ByVal **Style** As **OutlineStyle**) As **OutlineStyle**

Adds an outline style to the collection based on an existing style

Member of **OutlineStyles**

Parameters	Description
Style	Description of Style goes here (**in**)

Return value

Returns **OutlineStyle**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_OutlineStyles;FNC_AddStyle')} **Related Topics**

OutlineStyles

Class **OutlineStyles**

Properties Methods Referenced by

The **OutlineStyles** class defines the characteristics of outline collection objects and describes the look and behavior of the objects through its properties and methods. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

CorelDRAW comes with many different line styles. Line styles are preset lines that have different attributes, such as dotted lines, or dashed lines. Applying a line style does not change the shape of the line or the amount of the space it occupies on the drawing page. You can edit an existing line style to meet your needs.

{button ,AL(`CLS_OutlineStyles')} **Related Topics**

OutlineStyles.Application

Property **Application** AS **Application**

OutlineStyles

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub StylesApp()
MsgBox OutlineStyles.Application.Version
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Application')} **Related Topics**

OutlineStyles.Parent

Property **Parent** AS **Application**

OutlineStyles

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays a value associated with the shape type of the outline style's parent object:

```
Sub StylesParent()
MsgBox OutlineStyles.Parent.ActiveShape.Type
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Parent')} **Related Topics**

cdrNoShape=0

cdrRectangleShape=1

cdrEllipseShape=2

cdrCurveShape=3

cdrPolygonShape=4

cdrBitmapShape=5

cdrTextShape=6

cdrGroupShape=7

cdrSelectionShape=8

cdrGuidelineShape=9

cdrBlendGroupShape=10

cdrExtrudeGroupShape=11

cdrOLEObjectShape=12

cdrContourGroupShape=13

cdrLinearDimensionShape=14

cdrBevelGroupShape=15

cdrDropShadowGroupShape=16

cdr3DObjectShape=17

cdrArtisticMediaGroupShape=18

cdrConnectorShape=19

cdrMeshFillShape=20

OutlineStyles.Item

Property **Item**(ByVal **Index** AS Long) AS **OutlineStyle**

OutlineStyles

Description

The **Item** property returns a value associated with the index number of an outline style object in the **OutlineStyles** collection of CorelDRAW. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

`OutlineStyles(2)` refers to the second outline style object in the **OutlineStyles** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `OutlineStyles.Item(2)` is the same as `OutlineStyles(2)` - they both reference the second outline style object in the **OutlineStyles** collection.

You must reference an outline style in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in an **OutlineStyles** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the number of dash/gap pairs (the **DashCount** property) in the first outline style in the **OutlineStyles** collection:

```
Sub StylesItem()
MsgBox OutlineStyles.Item(1).DashCount
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Item')} **Related Topics**

OutlineStyles.Count

Property **Count** AS Long

OutlineStyles

Description

The **Count** property returns the number of outline styles in the **OutlineStyles** collection of CorelDRAW. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of outline styles in the **OutlineStyles** collection in a message box:

```
Sub StylesCount()
MsgBox OutlineStyles.Count
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Count')} **Related Topics**

OutlineStyles.Add

Function **Add**() AS **OutlineStyle**

OutlineStyles

Description

The **Add** method places an outline style in the **OutlineStyles** collection in CorelDRAW. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

Example

The following code example creates a new outline style and applies the new style to a line segment that is created on the active layer of the current document in CorelDRAW:

```
Sub StylesAdd()
Dim NewShape as Shape
Dim NewStyle as Style
Set NewStyle = OutlineStyles.Add     'creates a new style
NewStyle.DashCount = 1        '1 dash/gap pair
NewStyle.DashLength(1) = 20          'sets dash length
NewStyle.GapLength(1) = 1            'sets gap length
Set NewShape = ActiveLayer.CreateLineSegment (0, 0, 5, 5)
NewShape.Outline.Width = .1          'sets width of line
NewShape.Outline.Style = NewStyle    'sets outline style to newly created style
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Add')} **Related Topics**

OutlineStyles.Remove

Sub **Remove**(ByVal **Index** AS Long)

OutlineStyles

Description

The **Remove** method removes an outline style from the **OutlineStyles** collection in CorelDRAW. An outline is the area that surrounds an object. When you create an object, CorelDRAW gives it a default outline that is a black solid fill. You can specify the outline's color as well as set its width and style.

An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

It is important to note that removing an outline style from the collection will not affect any shapes in the document using the removed outline style.

In order to remove an outline style from the **OutlineStyles** collection, you must past the style's index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in a **OutlineStyles** collection; it uniquely identifies each member of the collection.

Example

The following code example removes the first outline style from the OutlineStyles collection in CorelDRAW:

```
Sub StylesRemove()
OutlineStyles.Remove (1)
End Sub
```

{button ,AL(`CLS_OutlineStyles;FNC_Remove')} **Related Topics**

ArrowHead properties

ArrowHead Legend

- Index

ArrowHead

Class **ArrowHead**

Properties Referenced by

The **ArrowHead** class defines the characteristics of arrowhead objects and describes the look and behavior of the objects through its properties and methods. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

{button ,AL(`CLS_ArrowHead')} **Related Topics**

ArrowHead.Index

Property **Index** AS Long

ArrowHead

Description

The **Index** property returns a value associated with an arrowhead object in the arrowheads collection of CorelDRAW. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shapes or a variety of other shapes. Each type of arrowhead in CorelDRAW is identified by an index number.

`ArrowHeads(5)` refers to the fifth arrowhead in the arrowheads collection of CorelDRAW.

The **Index** property returns a Read-Only value.

Example

The following code example takes all the shapes in the current selection, and determines which shapes have outlines. If the shape has an outline, an arrowhead with an idex of 5 replaces all the arrowheads that have an index of 1. This is done for both the start arrows and end arrows of the shape objects:

```
Sub Index()
Dim s As Shape
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
      With s.Outline
        If .StartArrow.Index = 1 Then
        .StartArrow = ArrowHeads(5)
        End If
        If s.Outline.EndArrow.Index = 1 Then
        s.Outline.EndArrow = ArrowHeads(5)
        End If
      End With
   End If
Next s
End If
End Sub
```

{button ,AL(`CLS_ArrowHead;FNC_Index')} **Related Topics**

ArrowHeads properties

ArrowHeads Legend

- Application

- Count

-
- Item

- Parent

ArrowHeads methods

ArrowHeads Legend

Remove

ArrowHeads

Class **ArrowHeads**

Properties Methods Referenced by

The **ArrowHeads** class defines the characteristics of ArrowHeads collection objects and describes the look and behavior of the collection objects through its properties and methods. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

{button ,AL(`CLS_ArrowHeads')} **Related Topics**

A collection is a group of objects that have similar characteristics and actions but are uniquely identified by index names or numbers. Collections are always plural (i.e. Arrowheads is a collection of arrowhead objects.)

ArrowHeads.Application

Property **Application** AS **Application**

ArrowHeads

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub VersionApplication()
With ActiveDocument.Selection.Shapes
    MsgBox "You are using CorelDRAW " & Version
End With
End Sub
```

{button ,AL(`CLS_ArrowHeads;FNC_Application')} **Related Topics**

ArrowHeads.Parent

Property **Parent** AS **Application**

ArrowHeads

Description

The **Parent** property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the active page in a message box:

```
Sub ArrowHeadsParent()
    MsgBox "The name of the ArrowHead object's parent is: " _
    & vbcrlf & vbcrlf & ArrowHeads.Parent.ActivePage.Name
End Sub
```

{button ,AL(`CLS_ArrowHeads;FNC_Parent')} **Related Topics**

ArrowHeads.Item

Property **Item**(ByVal **Index** AS Long) AS **ArrowHead**

ArrowHeads

Description

The **Item** property returns a value associated with the index number of an arrowhead object in the **ArrowHeads** collection of CorelDRAW. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

Each type of arrowhead in CorelDRAW is identified by an index number.

`ArrowHeads(5)` refers to the fifth arrowhead in the **ArrowHeads** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property of the ArrowHeads collection is the default property and may be omitted when referencing items in the collection. For example, `ArrowHeads.Item(5)` is the same as `ArrowHeads(5)` - they both reference the fifth arrowhead in the collection.

You must reference an arrowhead in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in an **ArrowHeads** collection; it uniquely identifies each member of the collection

Example

The following code example takes all the shapes in the current selection, and determines which shapes have outlines. If the shape has an outline, an arrowhead with an idex of 5 is applied to the start arrow position of each shape:

```
Sub Item()
Dim s As Shape
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
    s.Outline.StartArrow = ArrowHeads.Item(5)
'****can also be referenced as s.Outline.StartArrow = ArrowHeads (5)
    End If
Next s
End Sub
```

{button ,AL(`CLS_ArrowHeads;FNC_Item')} **Related Topics**

ArrowHeads.Count

Property **Count** AS Long

ArrowHeads

Description

The **Count** property returns the number of arrowheads in the **ArrowHeads** collection of CorelDRAW. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

The **Count** property returns a Read-Only value.

Example

The following code example identifies all arrowheads of selected objects that have an index of 1 and replaces them with arrowheads that have an index equal to the total number of arrowhead objects. This will replace the selected arrowheads with the last arrowhead in the ArrowHeads collection:

```
Sub Count()
Dim s As Shape
Dim Arrow As ArrowHead
Set Arrow = ArrowHeads(ArrowHeads.Count)
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
      With s.Outline
        If .StartArrow.Index = 1 Then
        .StartArrow = Arrow
        End If
        If .EndArrow.Index = 1 Then
        .StartArrow = Arrow
        End If
      End With
    End If
Next s
End Sub
```

{button ,AL(`CLS_ArrowHeads;FNC_Count')} **Related Topics**

ArrowHeads.Remove

Sub **Remove**(ByVal **Index** AS Long)

ArrowHeads

Description

The **Remove** method deletes an arrowhead object from the **ArrowHeads** collection in CorelDRAW. An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

You must reference an arrowhead in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each object in an **ArrowHeads** collection; it uniquely identifies each member of the collection

Example

The following code example removes the last arrowhead object from the **ArrowHeads** collection:

```
Sub ArrowHeadDelete()
ArrowHeads.Remove (ArrowHeads.Count)
End Sub
```

{button ,AL(`CLS_ArrowHeads;FNC_Remove')} **Related Topics**

FountainFill properties

FountainFill Legend

- Angle
  - BlendType
  - Colors

- EdgePad
  - EndColor
  - EndX
  - EndY
  - MidPoint
  - StartColor
  - StartX
  - StartY
  - Steps
  - Type

FountainFill methods

FountainFill Legend

- SetAngle
- SetEdgePad
- Translate

FountainFill

Class **FountainFill**

Properties Methods Referenced by

The **FountainFill** class defines the characteristics of fountain fill objects and describes the look and behavior of the objects through its properties and methods.

A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

{button ,AL(`CLS_FountainFill')} **Related Topics**

FountainFill.Type

Property **Type** AS **cdrFountainFillType**

FountainFill

Description

The **Type** property returns or sets a value that indicates a fountain fill's type in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

A fountain fill can be linear, conical, radial, or square. The **Type** parameter returns cdrFountainFillType.

Example

The following code example displays a value that indicates the fountain fill type of the active shape in a message box:

```
Sub FillType()
With ActiveShape.Fill.Fountain
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_Type')} **Related Topics**

FountainFill.StartX

Property **StartX** AS Double

FountainFill

Description

The **StartX** property returns or sets the horizontal start position of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

For radial, rectangular, and conical fills, the starting point is the center point of the fill.

The **StartX** property is measured in document units.

Example

The following code example displays the horizontal start position of the fountain fill effect in the active shape:

```
Sub Position()
With ActiveShape.Fill.Fountain
    MsgBox .StartX
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_StartX')} **Related Topics**

FountainFill.StartY

Property **StartY** AS Double

FountainFill

Description

The **StartY** property returns or sets the vertical start position of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

For radial, rectangular, and conical fills, the starting point is the center point of the fill.

The **StartY** property is measured in document units.

Example

The following code example displays the vertical start position of the fountain fill effect in the active shape:

```
Sub Position()
With ActiveShape.Fill.Fountain
    MsgBox .StartY
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_StartY')} **Related Topics**

FountainFill.EndX

Property **EndX** AS Double

FountainFill

Description

The **EndX** property returns or sets the horizontal end position of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

For radial, rectangular, and conical fills, the starting point is the center point of the fill.

The **EndX** property is measured in document units.

Example

The following code example displays the horizontal end position of the fountain fill effect in the active shape:

```
Sub Position()
With ActiveShape.Fill.Fountain
    MsgBox .EndX
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_EndX')} **Related Topics**

FountainFill.EndY

Property **EndY** AS Double

FountainFill

Description

The **EndY** property returns or sets the horizontal end position of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

For radial, rectangular, and conical fills, the starting point is the center point of the fill.

The **EndY** property is measured in document units.

Example

The following code example displays the vertical end position of the fountain fill effect in the active shape:

```
Sub Position()
With ActiveShape.Fill.Fountain
    MsgBox .EndY
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_EndY')} **Related Topics**

FountainFill.Angle

Property **Angle** AS Double

FountainFill

Description

The **Angle** property returns the degree measurement of the angle in linear, conical, or square fountain fill effects. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Changing the angle of gradation affects the slant of the fountain fill. Radial fountain fills progress in a series of concentric circles, so you can't change their angle. Positive values rotate the fill counterclockwise; negative values rotate it clockwise.

The **Angle** property returns a Read-Only value.

Example

The following code example sets the angle of the fountain fill effect in the active shape to 45 degrees and displays that value in a message box:

```
Sub AngleValue()
With ActiveShape.Fill.Fountain
    .SetAngle = 45
    MsgBox .Angle
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_Angle')} **Related Topics**

FountainFill.SetAngle

Sub **SetAngle**(ByVal **Angle** AS Double)

FountainFill

Description

The **Angle** property sets the degree measurement of the angle in linear, conical, or square fountain fill effects. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Changing the angle of gradation affects the slant of the fountain fill. Radial fountain fills progress in a series of concentric circles, so you can't change their angle. Positive values rotate the fill counterclockwise; negative values rotate it clockwise.

In order to use the **SetAngle** method, you must pass the **Angle** value as a parameter:

Parameters	Description
Angle	The **Angle** parameter specifies, in degrees, the size of the angle measurement in linear, conical, or square fountain fills. This value can range from 0 to 360.

Example

The following code example sets the angle of the fountain fill effect in the active shape to 45 degrees and displays that value in a message box:

```
Sub AngleValue()
With ActiveShape.Fill.Fountain
    .SetAngle = 45
    MsgBox .Angle
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_SetAngle')} **Related Topics**

FountainFill.Translate

Sub **Translate**(ByVal **X** AS Double, ByVal **Y** AS Double)

FountainFill

Description

The **Translate** method moves the start and end points of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

For radial, rectangular, and conical fills, the starting point is the center point of the fill. By moving the fountain fill, you are moving the X and Y coordinates of the fill's center point.

In order to use the **Translate** method, you must pass the **X** and **Y** values as parameters:

Parameters	Description
X	The **X** parameter specifies the horizontal coordinate of the center point in a radial, rectangular, and conical fountain fill. This value is measured in document units.
Y	The **Y** parameter specifies the vertical coordinate of the center point in a radial, rectangular, and conical fountain fill. This value is measured in document units.

Example

The following code example uses the Translate method to move the fountain fill effect in the active shape. The fountain fill's center point is given new coordinates, which change the appearance of the fill in the shape:

```
Sub FillMove()
With ActiveShape.Fill.Fountain
    .Translate 2, 4
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_Translate')} **Related Topics**

FountainFill.EdgePad

Property **EdgePad** AS Long

FountainFill

Description

The **EdgePad** parameter defines the length of solid colors at the beginning and end of the fountain fill before the start blending with the next color in the fountain fill. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors.

The **EdgePad** property returns a Read-Only value.

Example

The following code example sets the edge pad value in the fountain fill of the active shape to 25 percent and displays that value in a message box:

```
Sub Pad()
With ActiveShape.Fill.Fountain
    .SetEdgePad 25
    MsgBox .EdgePad
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_EdgePad')} **Related Topics**

FountainFill.Steps

Property **Steps** AS Long

FountainFill

Description

The **Steps** property returns or sets the number of steps used in a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

The **Steps** property identifies the number of bands (steps) used to display a fountain fill. When you create a fountain fill, the space required to blend the colors is divided by the number of fountain steps displayed in the Steps box. The Steps value can range from 2 to 999.

Example

The following code example sets the number of steps in the fountain fill effect of the active shape to 200:

```
Sub Step()
With ActiveShape.Fill.Fountain
    .Steps = 200
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_Steps')} **Related Topics**

FountainFill.BlendType

Property **BlendType** AS **cdrFountainFillBlendType**

FountainFill

Description

The **BlendType** property returns or sets the blend type in a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

The blend type defines how colors will appear with other colors in the fill effect. This value returns cdrFountainFillBlendType.

Example

The following code example sets the midpoint value to 50 for the fountain fill effect in the active shape. The blend type of the fountain fill is set to a direct fountain fill blend:

```
Sub MidPointValue()
With ActiveShape
    .BlendType = cdrDirectFountainFillBlend
    .MidPoint = 50
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_BlendType')} **Related Topics**

FountainFill.MidPoint

Property **MidPoint** AS Long

FountainFill

Description

The **MidPoint** property returns or sets the point where two colors converge in a two color fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

The MidPoint is an imaginary line between two colors in a fountain fill. The value of the mid-point represents its position in relation to two fountain fill colors. By changing this value, you can set the point at which two colors in a fountain fill converge. For example, in a two-color fountain fill using the colors black and white, a value of 50 positions the mid-point in the center of the fill so that half of the fill is black and half is white. Increasing the mid-point value to 99 results in a fountain fill dominated by black. Decreasing the mid-point value to 1 results in a fountain fill dominated by white.

Example

The following code example sets the midpoint value to 50 for the fountain fill effect in the active shape. The blend type of the fountain fill is set to a direct fountain fill blend:

```
Sub MidPointValue()
With ActiveShape
    .BlendType = cdrDirectFountainFillBlend
    .MidPoint = 50
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_MidPoint')} **Related Topics**

FountainFill.Colors

Property **Colors** AS **FountainColors**

FountainFill

Description

The **Colors** property returns or sets the **FountainColors** collection available to fountain fills in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

A fountain color is a specific color point used in a fountain fill effect in a shape object. A color point indicates a change from one color to the next in the fountain fill effect.

Example

The following code example displays the number of fountain colors available to the fountain fill effect in the active shape. A message box also displays the name of the start color and end color in the active fountain fill effect:

```
Sub Color()
With ActiveShape.Fill.Fountain
    MsgBox .Colors.Count
    MsgBox .StartColor.Name
    MsgBox .EndColor.Name
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_Colors')} **Related Topics**

FountainFill.StartColor

Property **StartColor** AS **Color**

FountainFill

Description

The **StartColor** property returns or sets the color that appears at the beginning of a fountain fill effect. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Example

The following code example displays the number of fountain colors available to the fountain fill effect in the active shape. A message box also displays the name of the start color and end color in the active fountain fill effect:

```
Sub Color()
With ActiveShape.Fill.Fountain
    MsgBox .Colors.Count
    MsgBox .StartColor.Name
    MsgBox .EndColor.Name
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_StartColor')} **Related Topics**

FountainFill.EndColor

Property **EndColor** AS **Color**

FountainFill

Description

The **EndColor** property returns or sets the color that appears at the end of a fountain fill effect. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Example

The following code example displays the number of fountain colors available to the fountain fill effect in the active shape. A message box also displays the name of the start color and end color in the active fountain fill effect:

```
Sub Color()
With ActiveShape.Fill.Fountain
    MsgBox .Colors.Count
    MsgBox .StartColor.Name
    MsgBox .EndColor.Name
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_EndColor')} **Related Topics**

FountainFill.SetEdgePad

Sub **SetEdgePad**(ByVal **EdgePad** AS Long)

FountainFill

Description

The **SetEdgePad** method sets the edge pad value of a fountain fill in CorelDRAW. The edge pad defines the length of solid colors at the beginning and end of the fountain fill before the start blending with the next color in the fountain fill. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors.

In order to use the **SetEdgePage** method, you must pass the **EdgePad** value as a parameter:

Parameters	Description
EdgePad	The **EdgePad** parameter defines the length of solid colors at the beginning and end of the fountain fill before the start blending with the next color in the fountain fill. You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors. This value is optional and the default value is 0. The EdgePad parameter is a percentage value and its value range is 0 to 49.

Example

The following code example sets the edge pad value in the fountain fill of the active shape to 25 percent and displays that value in a message box:

```
Sub Pad()
With ActiveShape.Fill.Fountain
    .SetEdgePad 25
    MsgBox .EdgePad
End With
End Sub
```

{button ,AL(`CLS_FountainFill;FNC_SetEdgePad')} **Related Topics**

FountainColor properties

FountainColor Legend

- Color
- Position

FountainColor methods

FountainColor Legend

- Delete
- Move

FountainColor

Class **FountainColor**

Properties Methods Referenced by

The **FountainColor** class defines the characteristics of fountain color point objects and describes the look and behavior of the collection objects through its properties and methods.

A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

{button ,AL(`CLS_FountainColor')} **Related Topics**

FountainColor.Position

Property **Position** AS Long

FountainColor

The **Position** property returns a value associated with the position of a fountain color point in the **FountainColors** collection. A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

If a color is given the position of 2, it is actually the third color point in the fountain effect, since the starting point holds an index of 0 and is not included as a collection point. Valid positions range from 1 to 99.

The **Position** property returns a Read-Only value.

Example

The following code example displays the position value of the color point in the **FountainColors** collection with an index of 2 This color point is actually the third point in the collection since the starting point, with an index of 0, is not included in the collection:

```
Sub FountainColorPosition()
With ActiveShape.Fill.Fountain.Colors
    MsgBox .Item(2).Position
End With
End Sub
```

{button ,AL(`CLS_FountainColor;FNC_Position')} **Related Topics**

FountainColor.Color

Property **Color** AS **Color**

FountainColor

Description

The **Color** property returns or sets a value associated with a fountain color point in the **FountainColors** collection. A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

Example

The following code example displays the color type of the color point in the **FountainColors** collection that has an index value of 1. This color point is actually the second point in the collection since the starting point, with an index of 0, is not included in the collection.

```
Sub FountainColorType()
With ActiveShape.Fill.Fountain.Colors
   MsgBox .Item(1).Type
End With
End Sub
```

{button ,AL(`CLS_FountainColor;FNC_Color')} **Related Topics**

FountainColor.Move

Sub **Move**(ByVal **NewPosition** AS Long)

FountainColor

Description

The **Move** method sets the position of a fountain color point in the **FountainColors** collection. A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

Moving a color point in the **FountainColors** collection with the **Move** method will replace an existing color point if the existing color point is positioned at the specified position point.

If a color is given the position of 2, it is actually the third color point in the fountain effect, since the starting point holds an index of 0 and is not included as a collection point. Valid positions range from 1 to 99.

Parameters	Description
NewPosition	The **NewPosition** parameter defines the new location of a color point in the **FountainColors** collection. If a color is given the position of 2, it is actually the third color point in the fountain effect, since the starting point holds an index of 0 and is not included as a collection point. Valid positions range from 1 to 99.

Example

The following code example moves the color point in the **FountainColors** collection with an index of 4. This color point is actually the fifth point in the collection since the starting point, with an index of 0, is not included in the collection. This color point is moved from its current position the 22nd position in the collection:

```
Sub FountainColorMove()
With ActiveShape.Fill.Fountain.Colors
    .Item(4).Move (22)
End With
End Sub
```

{button ,AL(`CLS_FountainColor;FNC_Move')} **Related Topics**

FountainColor.Delete

Sub **Delete**()

FountainColor

Description

The **Delete** method removes a fountain color point from the **FountainColors** collection in CorelDRAW. A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

It is important to note that you cannot delete starting and ending color points. These points hold index values of 0 and Count+1 respectively, and they are not included in the **FountainColors** collection.

Example

The following code example deletes the color point in the **FountainColors** collection with an index of 4. This color point is actually the fifth point in the collection since the starting point, with an index of 0, is not included in the collection. A message box displays the number of color points in the collection before and after the deletion, so you can see that the color point is deleted:

```
Sub FountainColorDelete()
With ActiveShape.Fill.Fountain.Colors
    MsgBox .Count
    .Item(4).Delete
    .MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_FountainColor;FNC_Delete')} **Related Topics**

FountainColors properties

FountainColors Legend

- Count
  GrayLevel
- Item

FountainColors methods

FountainColors Legend

- Add
- AddGrayLevel

A color mode that displays images using 256 shades of gray. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white). In the RGB color mode, a grayscale value corresponds to equal amounts of all RGB colors; in CMYK, a grayscale value corresponds to zero C, M, and Y values, with a positive K value; in HSB, a grayscale value corresponds to zero H and S values, with a positive B value. The Grayscale color mode is based on the Grayscale color model.

FountainColors

Class **FountainColors**

Properties Methods Referenced by

The **FountainColors** class defines the characteristics of **FountainColors** collection objects and describes the look and behavior of the collection objects through its properties and methods.

A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

{button ,AL(`CLS_FountainColors')} **Related Topics**

FountainColors.Item

Property **Item**(ByVal **Index** AS Long) AS **FountainColor**

FountainColors

Description

The **Item** property returns a value associated with the index number of a color point in the **FountainColors** collection of CorelDRAW.A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

`Fountain.Colors(2)` refers to the second color in the **FountainColors** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Fountain.Colors.Item(2)` is the same as `Fountain.Colors(2)` - they both reference the second color in the collection.

You must reference an color point in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each color in a **FountainColors** collection; it uniquely identifies each member of the collection. The **Index** value ranges from 0 to Count+1, where 0 is the starting point and Count+1 is the ending point in the collection.

Example

The following code example displays the name of the first color point in the **FountainColors** collection in a message box:

```
Sub FountainItem()

With ActiveShape.Fill.Fountain.Colors
    MsgBox .Item(2).Color.Name
End With
End Sub
```

{button ,AL(`CLS_FountainColors;FNC_Item')} **Related Topics**

FountainColors.Count

Property **Count** AS Long

FountainColors

Description

The **Count** property returns the number of colors in the **FountainColors** collection of CorelDRAW. A fountain color is a specific color point used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

It is important to note that the starting and ending color points in the FountainColors collection are not included in the count of collection color points. Only the color points between the start and end points are included with the **Count** property.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of colors in the **FountainColors** collection in a message box, excluding the starting and ending color points:

```
Sub FountainCount()
With ActiveShape.Fill.Fountain.Colors
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_FountainColors;FNC_Count')} **Related Topics**

FountainColors.Add

Sub **Add**(ByRef **Color** AS **Color**, ByVal **Position** AS Long)

FountainColors

Description

The **Add** method places a new color in the **FountainColors** collection of CorelDRAW. A fountain color is a specific color point that is used in a fountain fill effect in a shape object. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

Adding a color point to the fountain fill effect with the **Add** method will replace an existing color point if the existing color point is positioned at the specified position point.

In order to add a color point with the **Add** method, you must pass the **Color** and the **Position** of the new color as parameters:

Parameters	Description
Color	The **Color** parameter defines and existing or new color to be added to the **FountainColors** collection. It uses an existing color name or a **CreateColor** method (i.e. **CreateCMYKColor**, **CreateRGBColor**) to specify a color that is added to the collection.
Position	The **Position** parameter defines the location of the new color in the **FountainColors** collection. If a color is given the position of 2, it is actually the third color point in the fountain effect, since the starting point holds an index of 0 and is not included as a collection point. Valid positions range from 1 to 99.

Example

The following code example creates and adds a new color point to the **FountainColors** collection in CorelDRAW. The color is created using the **CreateCMYKColor** method and is given an index of 1, placing the new color first in the **FountainColors** collection, after the starting color point, which holds and index of 0:

```
Sub FountainAdd()
With ActiveShape.Fill.Fountain.Colors
    .Add CreateCMYKColor (0, 0, 50, 30), 1
End With
End Sub
```

{button ,AL(`CLS_FountainColors;FNC_Add')} **Related Topics**

FountainColors.GrayLevel

Property **GrayLevel**(ByVal **Index** AS Long) AS Long

FountainColors

Description

The **GrayLevel** property returns or sets a value associated with the gray component of a color in the **FountainColors** collection in CorelDRAW. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

The gray level is used to specify the transparency level for the fountain transparency. The Index value can range from 0 to Count+1, where 0 is the starting color point and Count+1 is the ending color point in the fountain or transparency.

You must reference the gray level by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each color point in a **FountainColors** collection; it uniquely identifies each member of the collection. The **Index** of the Gray Level will be the same as the **Position** parameter in the **Add** method of the **FountainColors** collection. The **Index** value ranges from 0 to Count+1, where 0 is the starting point and Count+1 is the ending point in the collection.

Example

The following code example displays the gray level value of the third gray level in the **FountainColors** collection in a message box:

```
Sub FountainGrey()
With ActiveShape.Fill.Fountain.Colors
    MsgBox .GrayLevel (3)
End With
End Sub
```

{button ,AL(`CLS_FountainColors;FNC_GrayLevel')} **Related Topics**

FountainColors.AddGrayLevel

Sub **AddGrayLevel**(ByVal **GrayLevel** AS Long, ByVal **Position** AS Long)

FountainColors

Description

The **AddGrayLevel** method adds a new grayscale color point to the **FountainColors** collection at a specified position in the collection. A fountain fill is a complex fill that displays a progression between two or more colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. A color point indicates a change from one color to the next in the fountain fill effect.

Adding a color point to the fountain fill effect with the **AddGrayLevel** method will replace an existing color point if the existing color point is positioned at the specified position point.

In order to use the **AddGrayLevel** method, you must pass the **GrayLevel** and the **Position** components of the new color as parameters:

Parameters	Description
GrayLevel	The gray level is used to specify the transparency level for the fountain transparency. The Index value can range from 0 to Count+1, where 0 is the starting color point and Count+1 is the ending color point in the fountain or transparency.
Position	The **Position** parameter defines the location of gray level in the **FountainColors** collection. If a color is given the position of 2, it is actually the third color point in the fountain effect, since the starting point holds an index of 0 and is not included as a collection point. Valid positions range from 1 to 99.

Example

The following code example adds a new gray level to the **FountainColors** collection in CorelDRAW. The new gray level has a value of 10, and the position of the gray level in the collection is the third position:

```
Sub FountainAddGray()
With ActiveShape.Fill.Fountain.Colors
    .AddGrayLevel 10, 3
End With
End Sub
```

{button ,AL(`CLS_FountainColors;FNC_AddGrayLevel')} **Related Topics**

PatternFill properties

PatternFill Legend

- BackColor
- Canvas
- FilePath
- FrontColor
- MirrorFill
- OriginX
- OriginY
- RotationAngle
- SkewAngle
- TileHeight
- TileOffset
- TileOffsetType
- TileWidth
- TransformWithShape
- Type

PatternFill methods

PatternFill Legend

Load

PatternFill

Class **PatternFill**

Properties Methods Referenced by

The **PatternFill** class defines the characteristics of pattern fill objects and describes the look and behavior of the objects through its properties and methods. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

In CorelDRAW you can choose from a two-color, full-color, or bitmap pattern fill. A two-color pattern is a simple picture composed of only "on" and "off" pixels. The only colors included in the bitmap are the two that you assign. A full-color pattern is a picture composed of lines and fills, instead of dots of color like a bitmap. These vector graphics are smoother and more complex than bitmap images and are easier to manipulate. A bitmap pattern is a regular color picture (like you might get with an electronic photograph). These bitmaps vary in complexity, and it is best to use less complex bitmaps for fill patterns, because complex ones are memory-intensive and slow to display. The complexity of a bitmap is determined by its size, resolution, and bit depth.

You can rotate, skew, adjust the tile size, and change the center of the pattern or texture to create a custom fill. You can adjust the pattern or texture fill using the on-screen fill tiling vector. The Transform Fill With Object option lets you perform transform an object and fill simultaneously.

{button ,AL(`CLS_PatternFill')} **Related Topics**

Vector graphics are images are stored as algebraic equations that define the lines and curves of the drawing. They can also include bitmap information. They are created in illustration applications, such as CorelDRAW, or bitmap tracing applications, such as Corel TRACE. Vector formats are not restricted to certain color depths.

Compare to bitmap images, which are created pixel-by-pixel in paint applications and by scanners.

cdrTwoColorPattern=0

cdrFullColorPattern=1

cdrBitmapPattern=2

PatternFill.MirrorFill

Property **MirrorFill** As Boolean

Description

The **MirrorFill** property returns or sets a True or False value that indicates whether or not to mirror a pattern fill effect in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

When you mirror an object, you flip the object horizontally, vertically, or diagonally.

Example

The following code example mirrors the pattern fill in the active shape of CorelDRAW:

```
Sub PatternType()
With ActiveShape.Fill.Pattern
    .MirrorFill = True
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_MirrorFill')} **Related Topics**

PatternFill.Type

Property **Type** AS **cdrPatternFillType**

PatternFill

Description

The **Type** property returns or sets the type of pattern fill in an object in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

The **Type** property returns a value of cdrPatternFillType.

Example

The following code example displays a numerical value associated with the pattern fill type in the active shape in a message box:

```
Sub PatternType()
With ActiveShape.Fill.Pattern
    MsgBox .Type
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_Type')} **Related Topics**

PatternFill.FrontColor

Property **FrontColor** AS **Color**

PatternFill

Description

The **FrontColor** property returns or sets the foreground color in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Example

The following code example displays the name of the foreground color used in the pattern fill of the active shape object:

```
Sub PatternFront()
With ActiveShape.Fill.Pattern
    MsgBox .FrontColor.Name
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_FrontColor')} **Related Topics**

PatternFill.BackColor

Property **BackColor** AS **Color**

PatternFill

Description

The **BackColor** property returns or sets the background color in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Example

The following code example displays the name of the background color used in the pattern fill of the active shape object:

```
Sub PatternBack()
With ActiveShape.Fill.Pattern
    MsgBox .BackColor.Name
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_BackColor')} **Related Topics**

PatternFill.Canvas

Property **Canvas** AS **PatternCanvas**

PatternFill

Description

The **Canvas** property returns or sets the pattern canvas in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

Example

The following code example applies a predefined pattern fill, overlaid with a checkerboard pattern, to an ellipse:

```
Sub CanvasPattern()
Dim s As Shape
Dim pf As PatternFill
Dim cnv As New PatternCanvas
cnv.PutCopy PatternCanvases(2)        'copies the second preset canvas pattern
Set s = ActiveLayer.CreateEllipse(8, 7, 3, 4)
    s.Fill.ApplyNoFill
Set pf = s.Fill.ApplyPatternFill(cdrTwoColorPattern, , , CreateColorEx(5005, 255, 0, 0), _
    CreateColorEx(5005, 255, 255, 0)) 'creates a new pattern fill
    pf.Canvas = cnv    'combines canvas and pattern to create a new fill effect
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_Canvas')} **Related Topics**

PatternFill.FilePath

Property **FilePath** AS String

PatternFill

Description

The **FilePath** property returns a string value associated with the computer location of a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

This property returns the path of the pattern fill within the CorelDRAW folder. For example, if the full path of the pattern fill is "C:\Corel\Graphics10\Custom\Patterns", the **FilePath** property returns only "Custom\Patterns".

The **FilePath** property returns a Read-Only value.

Example

The following code example displays the file path of the pattern fill in the active shape:

```
Sub PatternPath()
With ActiveShape.Fill.Pattern
    MsgBox .FilePath
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_FilePath')} **Related Topics**

PatternFill.OriginX

Property **OriginX** AS Double

PatternFill

Description

The **OriginX** property returns or sets the width of a pattern fill pattern in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

The **OriginX** property is measured in document units.

Example

The following code example sets **OriginX** and **OriginY** properties of the current pattern fill in the active shape of CorelDRAW. The values are measured in document units:

```
Sub PatternOrigins()
With ActiveShape.Fill.Pattern
    .OriginX = 2       'width of the pattern fill is 2 document units (i.e. inches)
    .OriginY = 2       'height of the pattern fill is 2 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_OriginX')} **Related Topics**

PatternFill.OriginY

Property **OriginY** AS Double

PatternFill

Description

The **OriginY** property returns or sets the height of a pattern fill pattern in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

The **OriginY** property is measured in document units.

Example

The following code example sets **OriginX** and **OriginY** properties of the current pattern fill in the active shape of CorelDRAW. The values are measured in document units:

```
Sub PatternOrigins()
With ActiveShape.Fill.Pattern
    .OriginX = 2       'width of the pattern fill is 2 document units (i.e. inches)
    .OriginY = 2       'height of the pattern fill is 2 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_OriginY')} **Related Topics**

PatternFill.TileWidth

Property **TileWidth** AS Double

PatternFill

Description

The **TileWidth** property returns or sets the width of a tile in a pattern fill pattern in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Setting the **TileWidth** and **TileHeight** properties alter the density of the pattern fill. If the values are decreased, the density increases in the pattern texture fill.

The height and width properties are measured in document units.

Example

The following code example sets the width and height of the pattern fill in the active shape of CorelDRAW. The measurements are in document units:

```
Sub PatternTile()
With ActiveShape.Fill.Pattern
    .TileWidth = 6     'each tile has a width of 6 document units (i.e. inches)
    .TileHeight = 5    'each tile has a height of 5 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_TileWidth')} **Related Topics**

PatternFill.TileHeight

Property **TileHeight** AS Double

PatternFill

Description

The **TileHeight** property returns or sets the height of a tile in a pattern fill pattern in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Setting the **TileWidth** and **TileHeight** properties alter the density of the pattern fill. If the values are decreased, the density increases in the pattern texture fill.

The height and width properties are measured in document units.

Example

The following code example sets the width and height of the pattern fill in the active shape of CorelDRAW. The measurements are in document units:

```
Sub PatternTile()
With ActiveShape.Fill.Pattern
    .TileWidth = 6     'each tile has a width of 6 document units (i.e. inches)
    .TileHeight = 5    'each tile has a height of 5 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_TileHeight')} **Related Topics**

PatternFill.TileOffsetType

Property **TileOffsetType** AS **cdrTileOffsetType**

PatternFill

Description

The **TileOffsetType** property returns or sets the offset type of a tile in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

A pattern may be offset by row or by column. Offsetting the tiles in a pattern fill or texture fill lets you specify exactly where the patterns or textures begin. When you adjust the horizontal or vertical position of the first pattern or texture, relative to the top of the object, your adjustment affects the rest of the pattern or texture. The Preview window reflects the changes of any offset.

The **TileOffsetType** property returns a value of cdrTileOffsetType.

Example

The following code example displays the offset type of the active shape's pattern fill in a message box. A pattern is offset by either its rows or its columns and returns a value of cdrTileOffsetType.

```
Sub PatternOffsetType()
With ActiveShape.Fill.Pattern
    MsgBox .TileOffsetType
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_TileOffsetType')} **Related Topics**

PatternFill.TileOffset

Property **TileOffset** AS Long

PatternFill

Description

The **TileOffset** property returns or sets a value associated with the degree of tile offset in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

A pattern may be offset by row or by column. Offsetting the tiles in a pattern fill or texture fill lets you specify exactly where the patterns or textures begin. When you adjust the horizontal or vertical position of the first pattern or texture, relative to the top of the object, your adjustment affects the rest of the pattern or texture. The Preview window reflects the changes of any offset.

The **TileOffset** property is a numerical value that indicates a percentage of the actual tile size in the pattern fill. If .TileOffset = 15, the offset value is 15% of the tile size.

Example

The following code example sets the offset type of the pattern fill in the active shape to a column offset. The tile offset is set to 20, which sets the offset tile to 20% of the actual tile size:

```
Sub PatternOffset()
With ActiveShape.Fill.Pattern
    .TileOffsetType = cdrTileOffsetColumn
    .TileOffset = 20
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_TileOffset')} **Related Topics**

PatternFill.SkewAngle

Property **SkewAngle** AS Double

PatternFill

Description

The **SkewAngle** property returns or sets a value associated with the degree of skew in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Skewing a pattern fill applies a slant to the fill object. The **SkewAngle** property returns or sets the degree of that slant in a pattern fill. If .SkewAngle = 45, the pattern fill is slanted 45 degrees.

Example

The following code example sets the **TransformWithShape** property to True. This allows the pattern fill to alter its appearance in order to fit a changing shape and maintains the scale of the pattern fill in relation to the size of the shape object. The rotation angle of the pattern fill is set to 45 degrees and the skew angle of the pattern fill is set to 10 degrees:

```
Sub PatternTransform()
With ActiveShape.Fill.Pattern
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_SkewAngle')} **Related Topics**

PatternFill.RotationAngle

Property **RotationAngle** AS Double

PatternFill

Description

The **RotationAngle** property returns or sets a value associated with the degree of rotation in a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Rotating a texture fill moves changes the direction of a pattern in the fill object. The **RotationAngle** property returns or sets the degree of that direction in a pattern fill. If .RotationAngle = 45, the pattern rotates by 45 degrees in the fill object.

Example

The following code example sets the **TransformWithShape** property to True. This allows the pattern fill to alter its appearance in order to fit a changing shape and maintains the scale of the pattern fill in relation to the size of the shape object. The rotation angle of the pattern fill is set to 45 degrees and the skew angle of the pattern fill is set to 10 degrees:

```
Sub PatternTransform()
With ActiveShape.Fill.Pattern
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_RotationAngle')} **Related Topics**

PatternFill.TransformWithShape

Property **TransformWithShape** AS Boolean

PatternFill

Description

The **TransformWithShape** property returns or sets a True or False value, indicating if the pattern fill changes to fit its shape when the shape object is altered in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

If the **TransformWithShape** property is set to True, the pattern fill will change according to changes in its shape object. If set to False, the pattern fill will remain the same, despite changes in the shape object's appearance.

Example

The following code example sets the **TransformWithShape** property to True. This allows the pattern fill to alter its appearance in order to fit a changing shape and maintains the scale of the pattern fill in relation to the size of the shape object. The rotation angle of the pattern fill is set to 45 degrees and the skew angle of the pattern fill is set to 10 degrees:

```
Sub PatternTransform()
With ActiveShape.Fill.Pattern
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_TransformWithShape')} **Related Topics**

PatternFill.Load

Function **Load**(ByVal **FileName** AS String) AS Boolean

PatternFill

Description

The **Load** method returns or sets a True or False value that imports a graphic to use as a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Imported patterns are added to the end of the list of preset patterns. They are not automatically applied to the selected shape object. The graphic is then tiled to form a pattern in a path to which it is applied. If the **Load** method is True, the graphic is imported into CorelDRAW to be used as a pattern fill.

In order to import a graphic with the **Load** method, you must pass the **FileName** value as a parameter:

Parameters	Description
FileName	The **FileName** parameter identifies the full path name of the graphic that is imported into CorelDRAW to be used as a pattern fill. A file name contains the computer's path and the name of the graphic file.

Example

The following code example imports a preset checkered pattern fill into CorelDRAW:

```
Sub PatternLoad()
With ActiveShape.Fill.Pattern
    .Load ("C:\Corel\Graphics10\Custom\Patterns\checker.pat")
End With
End Sub
```

{button ,AL(`CLS_PatternFill;FNC_Load')} **Related Topics**

PatternCanvas properties

PatternCanvas Legend

- Data
  - Height
- Index
  - Pixel
  - Size
  - Width

PatternCanvas methods

PatternCanvas Legend

- Clear
- CopyArea
- FillArea
- FlipArea
- Line
- PSet
- PutCopy
- RotateArea
- Select

PatternCanvas

Class **PatternCanvas**

Properties Methods Referenced by

The **PatternCanvas** class defines the characteristics of pattern canvas objects and describes the look and behavior of the objects through its properties and methods. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Customized or preset patterns are applied or added to pattern canvases to create new and different pattern images in CorelDRAW. A pattern canvas acts as a background effect to a pattern fill.

{button ,AL(`CLS_PatternCanvas')} **Related Topics**

cdrPatternCanvas16x16=0

cdrPatternCanvas32x32=1

cdrPatternCanvas64x64=2

cdrPatternCanvasCustom=3

PatternCanvas.Data

Property **Data** As String

Description

The **Data** property returns or sets the canvas data in CorelDRAW. Canvas data is a string value that provides descriptive information about the canvas object. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

Example

The following code example creates a new pattern canvas and places the data from an existing pattern canvas into the new canvas with the **PutCopy** method. The size of the new canvas, which returns a value of cdrPatternCanvasSize, is set and the data, height and width values display in a message box:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Size = cdrPatternCanvas16x16
    MsgBox .Height
    MsgBox .Width
    MsgBox .Data
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Data')} **Related Topics**

PatternCanvas.Size

Property **Size** AS **cdrPatternCanvasSize**

PatternCanvas

Description

The **Size** property returns or sets the size of the pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

The size of a pattern canvas is measured in pixels. A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

The **Size** property returns a value of cdrPatternCanvasSize.

Example

The following code example creates a new pattern canvas and places the data from an existing pattern canvas into the new canvas with the **PutCopy** method. The size of the new canvas, which returns a value of cdrPatternCanvasSize, is set and the data, height and width values display in a message box:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Size = cdrPatternCanvas16x16
    MsgBox .Height
    MsgBox .Width
    MsgBox .Data
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Size')} **Related Topics**

PatternCanvas.Pixel

Property **Pixel**(ByVal **X** AS Long, ByVal **Y** AS Long) AS Boolean

PatternCanvas

Description

The **Pixel** property returns or sets a True or False value that indicates the presence of pixels at a specified coordinate position on a pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

In order to return or use the pixel value, you must pass the **X** and **Y** values of the coordinate as parameters:

Parameters	Description
X	The **X** parameter specifies the horizontal position (using the CorelDRAW rulers as a reference) for a point on the pattern canvas. This value is measured in document units.
Y	The **Y** parameter specifies the vertical position (using the CorelDRAW rulers as a reference) for a point on the pattern canvas. This value is measured in document units.

Example

The following code example sets the pixel value to True at the point (3,3) in the new pattern canvas:

```
Sub Canvas()
Dim pf as New PatternFill
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
With NewCanvas
    .Pixel (3, 3) = True
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Pixel')} **Related Topics**

PatternCanvas.Index

Property **Index** AS Long

PatternCanvas

Description

The **Index** property returns the index number of a pattern canvas in the PatternCanvases collection of CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. The **Index** property is a numerical value that uniquely identifies a pattern canvas in CorelDRAW.

The **Index** property returns a Read-Only value.

Example

The following code example uses the **PutCopy** method to copy data from an existing pattern canvas into a new pattern canvas. The existing pattern canvas is identified by its index number:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Index')} **Related Topics**

PatternCanvas.Line

Sub **Line**(ByVal **Flags** As Integer, ByVal **x1** As Long, ByVal **y1** As Long, ByVal **x2** As Long, ByVal **y2** As Long, ByVal **Color** As Boolean)

Description

The **Line** method draws a line on a pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

In order to return or use the pixel value, you must pass several values as parameters:

Parameters	Description
Flags	Description of Flags goes here (**in**)
x1	The **x1** parameter defines the horizontal, or X, coordinate for the start point of the new line on the canvas. This value is measured in document units.
y1	The **y1** parameter defines the vertical, or Y, coordinate for the start point of the new line on the canvas. This value is measured in document units.
x2	The **x2** parameter defines the horizontal, or X, coordinate for the end point of the new line on the canvas. This value is measured in document units.
y2	The **y2** parameter defines the vertical, orY, coordinate for the end point of the new line on the canvas. This value is measured in document units.
Color	The **Color** parameter is a True or False value that indicates if the new line has a color applied to it on the pattern canvas. If this value is set to True, the selected color in the active color palette is applied to the new line.

{button ,AL(`CLS_PatternCanvas;FNC_Line')} **Related Topics**

PatternCanvas.PSet

Sub **PSet**(ByVal **Step** As Integer, ByVal **X** As Long, ByVal **Y** As Long, ByVal **Color** As Boolean)

Description

The **PSet** method sets a pixel value at a specified coordinate position on a pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

In order to return or use the pixel value, you must pass several values as parameters:

Parameters	Description
Step	The **Steps** parameter identifies the number of bands (steps).
X	The **X** parameter specifies the horizontal position (using the CorelDRAW rulers as a reference) for a point on the pattern canvas. This value is measured in document units.
Y	The **Y** parameter specifies the vertical position (using the CorelDRAW rulers as a reference) for a point on the pattern canvas. This value is measured in document units.
Color	The **Color** parameter is a True or False value that indicates if the pixel has a color applied to it on the pattern canvas. If this value is set to True, the selected color in the active color palette is applied to the pixel.

{button ,AL(`CLS_PatternCanvas;FNC_PSet')} **Related Topics**

PatternCanvas.FillArea

Sub **FillArea**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal State AS Boolean)

PatternCanvas

Description

The **FillArea** method fills an area of a pattern canvas defined by two coordinates in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. This method is only applicable to custom canvases - it will not work with preset CorelDRAW canvases.

In order use the **FillArea** method, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter specifies the horizontal, or X, coordinate for the first point that defines the area to fill on the pattern canvas. This value is measured in document units.
y1	The **y1** parameter specifies the vertical, or Y, coordinatefor the first point that defines the area to fill on the pattern canvas. This value is measured in document units..
x2	The **x1** parameter specifies the horizontal, or X, coordinatefor the second point that defines the area to fill on the pattern canvas. This value is measured in document units..
y2	The **y2** parameter specifies the vertical, or Y, coordinatefor the second point that defines the area to fill on the pattern canvas. This value is measured in document units..
State	The **State** parameter is a True or False value that indicates if the filled area is returned to its original state after the fill is removed on the pattern canvas. If the value is True, the area is returned to its original state. This value is optional and the default value is True.

Example

The following code example creates a new pattern canvas and pattern fill and applies is to the active shape in CorelDRAW. Several areas of the canvas are copied, filled, flipped, and rotated using the **CopyArea**, **FillArea**, **FlipArea**, and **RotateArea** methods:

```
Sub Canvas()
Dim pf as New PatternFill
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
With NewCanvas
    .CopyArea 1, 1, 2, 2, 5, 5
    'copies area and moves it
    .FillArea 2, 2, 3, 3, True
    'fills area, returns original state
    .FlipArea 1, 1, 3, 3, cdrFlipHorizontal
    'flips area horizontally
    .RotateArea 3, 3, 4, 4, 45
    'rotates area 45 degrees
End With
Set pf = ActiveShape.Fill.ApplyPatternFill (cdrTwoColorPattern, , , _
CreateColorEx (2, 255, 255, 0), CreateColorEx (2, 255, 0, 0))
'applies a yellow and red pattern
ActiveShape.Fill.Pattern.Canvas = NewCanvas
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_FillArea')} **Related Topics**

PatternCanvas.CopyArea

Sub **CopyArea**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **X** AS Long, ByVal **Y** AS Long)

PatternCanvas

Description

The **CopyArea** method copies an area of a pattern canvas, defined by two coordinates, and moves the area to a new location on the canvas. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. This method is only applicable to custom canvases - it will not work with preset CorelDRAW canvases.

In order to use the **CopyArea** method, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter specifies the horizontal, or X, coordinate for the first point that defines the area to copy on the pattern canvas. This value is measured in document units.
y1	The **y1** parameter specifies the vertical, or Y, coordinatefor the first point that defines the area to copy on the pattern canvas. This value is measured in document units..
x2	The **x1** parameter specifies the horizontal, or X, coordinatefor the second point that defines the area to copy on the pattern canvas. This value is measured in document units..
y2	The **y2** parameter specifies the vertical, or Y, coordinatefor the second point that defines the area to copy on the pattern canvas. This value is measured in document units..
X	The **X** parameter specifies the horizontal position (using the CorelDRAW rulers as a reference) for the point on the pattern canvas that marks the new location of the copied area. This value is measured in document units.
Y	The **Y** parameter specifies the vertical position (using the CorelDRAW rulers as a reference) for a point on the pattern canvas that marks the new location of the copied area. This value is measured in document units.

Example

The following code example creates a new pattern canvas and pattern fill and applies is to the active shape in CorelDRAW. Several areas of the canvas are copied, filled, flipped, and rotated using the **CopyArea**, **FillArea**, **FlipArea**, and **RotateArea** methods:

```
Sub Canvas()
Dim pf as New PatternFill
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
With NewCanvas
    .CopyArea 1, 1, 2, 2, 5, 5
    'copies area and moves it
    .FillArea 2, 2, 3, 3, True
    'fills area, returns original state
    .FlipArea 1, 1, 3, 3, cdrFlipHorizontal
    'flips area horizontally
    .RotateArea 3, 3, 4, 4, 45
    'rotates area 45 degrees
End With
Set pf = ActiveShape.Fill.ApplyPatternFill (cdrTwoColorPattern, , , _
CreateColorEx (2, 255, 255, 0), CreateColorEx (2, 255, 0, 0))
'applies a yellow and red pattern
ActiveShape.Fill.Pattern.Canvas = NewCanvas
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_CopyArea')} **Related Topics**

PatternCanvas.FlipArea

Sub **FlipArea**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **Axes** AS **cdrFlipAxes**)

PatternCanvas

Description

The **FlipArea** method repositions an area of a pattern canvas,defined by two coordinates,. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. This method is only applicable to custom canvases - it will not work with preset CorelDRAW canvases.

When you use the **FlipArea** method, you can flip the area horizontally, vertically or both. In order to flip an area of the canvas, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter specifies the horizontal, or X, coordinate for the first point that defines the area to flip on the pattern canvas. This value is measured in document units.
y1	The **y1** parameter specifies the vertical, or Y, coordinatefor the first point that defines the area to flip on the pattern canvas. This value is measured in document units..
x2	The **x1** parameter specifies the horizontal, or X, coordinatefor the second point that defines the area to flip on the pattern canvas. This value is measured in document units..
y2	The **y2** parameter specifies the vertical, or Y, coordinatefor the second point that defines the area to flip on the pattern canvas. This value is measured in document units..
Axes	The **Axes** parameter specifies how the area of the pattern canvas is flipped. An area can be flipped horizontally, vertically, or both. This value returns cdrFlipAxes.

Example

The following code example creates a new pattern canvas and pattern fill and applies is to the active shape in CorelDRAW. Several areas of the canvas are copied, filled, flipped, and rotated using the **CopyArea**, **FillArea**, **FlipArea**, and **RotateArea** methods:

```
Sub Canvas()
Dim pf as New PatternFill
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
With NewCanvas
    .CopyArea 1, 1, 2, 2, 5, 5
    'copies area and moves it
    .FillArea 2, 2, 3, 3, True
    'fills area, returns original state
    .FlipArea 1, 1, 3, 3, cdrFlipHorizontal
    'flips area horizontally
    .RotateArea 3, 3, 4, 4, 45
    'rotates area 45 degrees
End With
Set pf = ActiveShape.Fill.ApplyPatternFill (cdrTwoColorPattern, , , _
CreateColorEx (2, 255, 255, 0), CreateColorEx (2, 255, 0, 0))
'applies a yellow and red pattern
ActiveShape.Fill.Pattern.Canvas = NewCanvas
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_FlipArea')} **Related Topics**

PatternCanvas.RotateArea

Sub **RotateArea**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **Angle** AS Double)

PatternCanvas

Description

The **RotateArea** method rotates an area of a pattern canvas,defined by two coordinates,. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. This method is only applicable to custom canvases - it will not work with preset CorelDRAW canvases.

Rotating an object means to reposition and reorient an object by turning it around its center of rotation. The amount of turn the area rotates is measured in degrees.

When you use the **RotateArea** method, you must pass several values as parameters:

Parameters	Description
x1	The **x1** parameter specifies the horizontal, or X, coordinate for the first point that defines the area to rotate on the pattern canvas. This value is measured in document units.
y1	The **y1** parameter specifies the vertical, or Y, coordinatefor the first point that defines the area to rotate on the pattern canvas. This value is measured in document units..
x2	The **x1** parameter specifies the horizontal, or X, coordinatefor the second point that defines the area to rotate on the pattern canvas. This value is measured in document units..
y2	The **y2** parameter specifies the vertical, or Y, coordinatefor the second point that defines the area to rotate on the pattern canvas. This value is measured in document units..
Angle	The **Angle** parameter specifies the angle, measured in degrees, that the pattern canvas area is rotated. Angle = 45 rotates the node range 45.

Example

The following code example creates a new pattern canvas and pattern fill and applies is to the active shape in CorelDRAW. Several areas of the canvas are copied, filled, flipped, and rotated using the **CopyArea**, **FillArea**, **FlipArea**, and **RotateArea** methods:

```
Sub Canvas()
Dim pf as New PatternFill
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
With NewCanvas
    .CopyArea 1, 1, 2, 2, 5, 5
    'copies area and moves it
    .FillArea 2, 2, 3, 3, True
    'fills area, returns original state
    .FlipArea 1, 1, 3, 3, cdrFlipHorizontal
    'flips area horizontally
    .RotateArea 3, 3, 4, 4, 45
    'rotates area 45 degrees
End With
Set pf = ActiveShape.Fill.ApplyPatternFill (cdrTwoColorPattern, , , _
CreateColorEx (2, 255, 255, 0), CreateColorEx (2, 255, 0, 0))
'applies a yellow and red pattern
ActiveShape.Fill.Pattern.Canvas = NewCanvas
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_RotateArea')} **Related Topics**

PatternCanvas.Select

Sub **Select**(ByVal **Index** AS Long)

PatternCanvas

Description

The **Select** method chooses a specific pattern canvas from the **PatternCanvases** collection in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

A pattern canvas is identified by its index number. In order to select an canvas with the **Select** method, you must pass the index number of a canvas as a parameter:

Parameters	Description
Index	The **Index** parameter is a numerical value that uniquely identifies a pattern canvas in the **PatternCanvases** collection of CorelDRAW.

Example

The following code example selects the tenth pattern canvas in the **PatternCanvases** collection, using the **Select** method:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Select (10)
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Select')} **Related Topics**

PatternCanvas.Clear

Sub **Clear**()

PatternCanvas

Description

The **Clear** method removes all effects from a pattern canvas in CorelDRAW, creating a blank canvas. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

Example

The following code example removes a custom pattern canvas with the **Clear** method::

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Clear
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Clear')} **Related Topics**

PatternCanvas.PutCopy

Sub **PutCopy**(ByRef **PatternCanvas** AS **PatternCanvas**)

PatternCanvas

Description

The **PutCopy** method places the data from another pattern canvas into the active pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

Canvas data is a string value that provides descriptive information about the canvas object.

In order to use the **PutCopy** method, you must pass the **PatternCanvas** value as a parameter:

Parameters	Description
PatternCanvas	The **PatternCanvas** parameter specifies a pattern canvas object in CorelDRAW. The data of this canvas is copied with the PutCopy method, and placed within the current canvas. Customized or preset patterns are applied or added to pattern canvases to create new and different pattern images in CorelDRAW. A pattern canvas acts as a background effect to a pattern fill.

Example

The following code example creates a new pattern canvas and places the data from an existing pattern canvas into the new canvas with the **PutCopy** method. The size of the new canvas, which returns a value of cdrPatternCanvasSize, is set and the data, height and width values display in a message box:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Size = cdrPatternCanvas16x16
    MsgBox .Height
    MsgBox .Width
    MsgBox .Data
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_PutCopy')} **Related Topics**

PatternCanvas.Width

Property **Width** AS Long

PatternCanvas

Description

The **Width** property returns or sets a numerical value that indicates the width of a pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

The width of a pattern canvas is measured in document units.

Example

The following code example creates a new pattern canvas and places the data from an existing pattern canvas into the new canvas with the **PutCopy** method. The size of the new canvas, which returns a value of cdrPatternCanvasSize, is set and the data, height and width values display in a message box:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Size = cdrPatternCanvas16x16
    MsgBox .Height
    MsgBox .Width
    MsgBox .Data
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Width')} **Related Topics**

PatternCanvas.Height

Property **Height** AS Long

PatternCanvas

Description

The **Height** property returns or sets a numerical value that indicates the height of a pattern canvas in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

The height of a pattern canvas is measured in document units.

Example

The following code example creates a new pattern canvas and places the data from an existing pattern canvas into the new canvas with the **PutCopy** method. The size of the new canvas, which returns a value of cdrPatternCanvasSize, is set and the data, height and width values display in a message box:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
'data from the fifth canvas in the collection is copied to the new canvas
With NewCanvas
    .Size = cdrPatternCanvas16x16
    MsgBox .Height
    MsgBox .Width
    MsgBox .Data
End With
End Sub
```

{button ,AL(`CLS_PatternCanvas;FNC_Height')} **Related Topics**

TextureFill properties

TextureFill Legend

- LibraryName
  - MaximumTileWidth
  - MirrorFill
  - OriginX
  - OriginY
  - Properties
  - Resolution
  - RotationAngle
  - SkewAngle
- StyleName
- TextureName
  - TileHeight
  - TileOffset
  - TileOffsetType
  - TileWidth
  - TransformWithShape

TextureFill methods

TextureFill Legend

- Select
- SetProperties

TextureFill

Class **TextureFill**

Properties Methods Referenced by

The **TextureFill** class defines the characteristics of texture fill objects and describes the look and behavior of the objects through its properties and methods. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Texture fills, unlike tiled bitmap fills, fill a designated area with one image instead of with a series of repeating images. Texture fills increase the size of a file and the time it takes to print. Therefore, you may want to use these fills sparingly, especially with larger objects.

You can use colors from any color model or palette for texture fills. Since texture fills can only hold RGB colors, however, this can cause a color shift when you display or print the files.

Corel TEXTURE lets you design bitmap texture fills and modify preset textures. You can recreate the natural textures of wood, clouds, stone, ripples, waves, and wrinkles, or create artificial patterns such as checkers, dots, lines, and swirls. Corel TEXTURE lets you control lighting, design, color combinations, and gradations. The Texture wizard guides you through the process, or you can start with a blank texture.

{button ,AL(`CLS_TextureFill')} **Related Topics**

TextureFill.MirrorFill

Property **MirrorFill** As Boolean

Description

The MirrorFill property returns or sets a True or False value that indicates if a texture fill is mirrored in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Mirror means to flip an object horizontally, vertically, or diagonally.

Example

The following code example mirrors the texture fill in the active shape of CorelDRAW:

```
Sub TextureOrigins()
With ActiveShape.Fill.Texture
    .MirrorFill = True
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_MirrorFill')} **Related Topics**

TextureFill.OriginX

Property **OriginX** AS Double

TextureFill

Description

The **OriginX** property returns or sets the width of a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

The **OriginX** property is measured in document units.

Example

The following code example sets **OriginX** and **OriginY** properties of the current texture fill pattern in the active shape of CorelDRAW. The values are measured in document units:

```
Sub TextureOrigins()
With ActiveShape.Fill.Texture
    .OriginX = 2       'width of the texture is 2 document units (i.e. inches)
    .OriginY = 2       'height of the texture is 2 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_OriginX')} **Related Topics**

TextureFill.OriginY

Property **OriginY** AS Double

TextureFill

Description

The **OriginY** property returns or sets the height of a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

The **OriginY** property is measured in document units.

Example

The following code example sets **OriginX** and **OriginY** properties of the current texture fill pattern in the active shape of CorelDRAW. The values are measured in document units:

```
Sub TextureOrigins()
With ActiveShape.Fill.Texture
    .OriginX = 2       'width of the texture is 2 document units (i.e. inches)
    .OriginY = 2       'height of the texture is 2 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_OriginY')} **Related Topics**

TextureFill.TileWidth

Property **TileWidth** AS Double

TextureFill

Description

The **TileWidth** property returns or sets the width of a tile in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Setting the **TileWidth** and **TileHeight** properties alter the density of the pattern texture fill. If the values are decreased, the density increases in the pattern texture fill. The height and width properties are measured in document units.

Example

The following code example sets the width and height of the texture fill pattern in the active shape of CorelDRAW. The measurements are in document units:

```
Sub TextureTile()
With ActiveShape.Fill.Texture
    .TileWidth = 6     'each tile has a width of 6 document units (i.e. inches)
    .TileHeight = 5    'each tile has a height of 5 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TileWidth')} **Related Topics**

TextureFill.TileHeight

Property **TileHeight** AS Double

TextureFill

Description

The **TileHeight** property returns or sets the height of a tile in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Setting the **TileWidth** and **TileHeight** properties alter the density of the pattern texture fill. If the values are decreased, the density increases in the pattern texture fill. The height and width properties are measured in document units.

Example

The following code example sets the width and height of the texture fill pattern in the active shape of CorelDRAW. The measurements are in document units:

```
Sub TextureTile()
With ActiveShape.Fill.Texture
    .TileWidth = 6     'each tile has a width of 6 document units (i.e. inches)
    .TileHeight = 5    'each tile has a height of 5 document units (i.e. inches)
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TileHeight')} **Related Topics**

cdrTileOffsetRow=0

cdrTileOffsetColumn=1

TextureFill.TileOffsetType

Property **TileOffsetType** AS **cdrTileOffsetType**

TextureFill

Description

The **TileOffsetType** property returns or sets the offset type of a tile in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Offsetting the tiles in a pattern fill or texture fill lets you specify exactly where the patterns or textures begin. When you adjust the horizontal or vertical position of the first pattern or texture, relative to the top of the object, your adjustment affects the rest of the pattern or texture. The Preview window reflects the changes of any offset.

A texture fill pattern may be offset by row or by column. The **TileOffsetType** property returns a value of cdrTileOffsetType.

Example

The following code example displays the offset type of the active shape's texture fill pattern in a message box. A pattern is offset by either its rows or its columns and returns a value of cdrTileOffsetType:

```
Sub TextureOffsetType()
With ActiveShape.Fill.Texture
    MsgBox .TileOffsetType
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TileOffsetType')} **Related Topics**

TextureFill.TileOffset

Property **TileOffset** AS Long

TextureFill

Description

The **TileOffset** property returns or sets a value associated with the degree of tile offset in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Offsetting the tiles in a pattern fill or texture fill lets you specify exactly where the patterns or textures begin. When you adjust the horizontal or vertical position of the first pattern or texture, relative to the top of the object, your adjustment affects the rest of the pattern or texture. The Preview window reflects the changes of any offset.

The **TileOffset** property is a numerical value that indicates a percentage of the actual tile size in the texture fill pattern.
If `.TileOffset = 15`, the offset value is 15% of the tile size.

Example

The following code example sets the offset type of the texture in the active shape to a column offset. The tile offset is set to 20, which sets the offset tile to 20% of the actual tile size:

```
Sub TextureOffset()
With ActiveShape.Fill.Texture
    .TileOffsetType = cdrTileOffsetColumn
    .TileOffset = 20
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TileOffset')} **Related Topics**

TextureFill.SkewAngle

Property **SkewAngle** AS Double

TextureFill

Description

The **SkewAngle** property returns or sets a value associated with the degree of skew in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Skewing a texture fill applies a slant to the fill object. The **SkewAngle** property returns or sets the degree of that slant in a texture fill pattern. If `.SkewAngle = 45`, the texture pattern is slanted 45 degrees.

Example

The following code example sets the **TransformWithShape** property to True. This allows the texture to alter its appearance in order to fit a changing shape and maintains the scale of the texture in relation to the size of the shape object. The rotation angle of the texture is set to 45 degrees and the skew angle of the texture is set to 10 degrees:

```
Sub TextureTransform()
With ActiveShape.Fill.Texture
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_SkewAngle')} **Related Topics**

TextureFill.RotationAngle

Property **RotationAngle** AS Double

TextureFill

Description

The **RotationAngle** property returns or sets a value associated with the degree of rotation in a texture pattern fill in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Rotating a texture fill moves changes the direction of a texture pattern in the fill object. The **RotationAngle** property returns or sets the degree of that direction in a texture fill pattern. If `.RotationAngle = 45`, the texture pattern rotates by 45 degrees in the fill object.

Example

The following code example sets the **TransformWithShape** property to True. This allows the texture to alter its appearance in order to fit a changing shape and maintains the scale of the texture in relation to the size of the shape object. The rotation angle of the texture is set to 45 degrees and the skew angle of the texture is set to 10 degrees:

```
Sub TextureTransform()
With ActiveShape.Fill.Texture
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_RotationAngle')} **Related Topics**

TextureFill.TransformWithShape

Property **TransformWithShape** AS Boolean

TextureFill

Description

The **TransformWithShape** property returns or sets a True or False value, indicating if the texture fill pattern changes to fit its shape when the shape object is altered in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

If the **TransformWithShape** property is set to True, the texture fill pattern will change according to changes in its shape object. If set to False, the texture fill pattern will remain the same, despite changes in the shape object's appearance.

Example

The following code example sets the **TransformWithShape** property to True. This allows the texture to alter its appearance in order to fit a changing shape and maintains the scale of the texture in relation to the size of the shape object. The rotation angle of the texture is set to 45 degrees and the skew angle of the texture is set to 10 degrees:

```
Sub TextureTransform()
With ActiveShape.Fill.Texture
    .RotationAngle = 45
    .SkewAngle - 10
    .TransformWithShape = True
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TransformWithShape')} **Related Topics**

TextureFill.Resolution

Property **Resolution** AS Long

TextureFill

Description

The **Resolution** property returns or sets a value indicating the bitmap resolution of a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Resolution refers to the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size. The texture fill pattern is treated as a bitmap and its resolution determines the DPI value in the texture fill pattern.

DPI is a measure of a printer's resolution in dots per inch. Typical desktop laser printers print at 300 dpi. Image setters print at 1270 or 2540 dpi. Printers with higher dpi capabilities produce smoother and cleaner output. The term dpi is also used to measure scanning resolution and to indicate bitmap resolution.

Example

The following code example set the resolution of the texture in the active shape to 400 dpi (dots per inch). The higher the resolution, the better your image will appear when printed:

```
Sub TextureResolution()
With ActiveShape.Fill.Texture
    .Resolution = 400
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_Resolution')} **Related Topics**

TextureFill.MaximumTileWidth

Property **MaximumTileWidth** AS Long

TextureFill

Description

The **MaximumTileWidth** property returns or sets a numerical value that indicates the maximum tile width for a tile in a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

The **MaximumTileWidth** property is always measured in pixels. A pixel is an abbreviation for picture element. Pixels are dots on a computer or television screen that combine to form an image. Computer images are created as an array of pixels, each having a specific color.

Example

The following code example sets the maximum tile width in the texture of the active shape to 1200 pixels. Once this property has been set, the tile width, in any measurement, cannot exceed the maximum width equivalency set with the **MaximumTileWidth** property:

```
Sub TextureMaxTile()
With ActiveShape.Fill.Texture
    .MaximumTileWidth = 1200  'measurement always in pixels
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_MaximumTileWidth')} **Related Topics**

TextureFill.LibraryName

Property **LibraryName** AS String

TextureFill

Description

The **LibraryName** property returns a string value associated with the name of a texture fill pattern library in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

A texture library is a collection of texture fill pattern files in CorelDRAW. The **LibraryName** property returns the name of that texture library. There are several texture libraries included in CorelDRAW.

The **LibraryName** property returns a Read-Only value.

Example

The following code example displays the library name of a texture fill pattern in the active shape of CorelDRAW:

```
Sub TextureLibrary()
With ActiveShape.Fill.Texture
    MsgBox .LibraryName
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_LibraryName')} **Related Topics**

TextureFill.TextureName

Property **TextureName** AS String

TextureFill

Description

The **TextureName** property returns a string value associated with the name of a texture fill pattern in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

The **TextureName** property returns a Read-Only value.

Example

The following code example displays the name of a texture fill pattern in the active shape of CorelDRAW:

```
Sub NameTexture()
With ActiveShape.Fill.Texture
    MsgBox .TextureName
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_TextureName')} **Related Topics**

TextureFill.StyleName

Property **StyleName** AS String

TextureFill

Description

The **StyleName** property returns a string value associated with the name of a texture style in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

The **StyleName** property returns the name of a specific style of textures, not a particular texture name. For example, the style **Sky 5 Colors** contains several textures, including the texture **Heavenly Clouds**. A texture can have only one style but a style may contain many textures.

The **StyleName** property returns a Read-Only value.

Example

The following code example displays the style name of a texture fill pattern in the active shape of CorelDRAW:

```
Sub TextureStyle()
With ActiveShape.Fill.Texture
    MsgBox .StyleName
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_StyleName')} **Related Topics**

TextureFill.Properties

Property **Properties** As **TextureFillProperties**

TextureFill

Description

The **Properties** property returns or sets a texture fill pattern property in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Each texture fill pattern has a range of properties and some textures have more properties than others. For example, the texture Heavenly Clouds has 9 properties and Midday Clouds has 6 properties.

In order to set or return a property, you must pass the index number or name of the property as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each texture fill's property, it uniquely identifies each property. For example, **Texture #** is the first property of the **Heavenly Clouds** texture. It has an index of 1. **Name** is the unique string value that identifies each texture property. For example, **Texture #** identifies the first property of the **Heavenly Clouds** texture.

Example

The following code example uses the **Select** method to apply the **Air Brush** texture from the **Samples** library to the active shape in CorelDRAW. A message box displays the name of the ninth property for the Air Brush texture, which is the Brightness property. The brightness of the texture is set to 50%. This is reduce the brightness of the texture in the active shape by half:

```
Sub TextureProperties()
With ActiveShape.Fill.Texture
    .Select "Air Brush", "Samples"
    MsgBox .GetPropertiesName(9)
    .Properties(9) = 50
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_Properties')} **Related Topics**

TextureFill.Select

Sub **Select**(ByVal **Texture** AS String, ByVal Library AS String)

TextureFill

Description

The **Select** method selects a texture fill pattern from a specified library and applies that texture to an object in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

In order to select texture fill patterns with the **Select** method, you must pass the texture and library as parameters:

Parameters	Description
Texture	**Texture** is a fill that looks like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change. When you identify a texture fill pattern, you pass the Name of the texture as a parameter. For example, **Air Brush** is a texture name in CorelDRAW.
Library	A texture **Library** is a collection of texture files in CorelDRAW. When you identify a Library, you pass the name of the library as a parameter. For example, **Samples** is a texture library in CorelDRAW.

Example

The following code example uses the **Select** method to apply the **Air Brush** texture from the **Samples** library to the active shape in CorelDRAW. The **GetPropertiesCount** method displays the number of properties associated with the Air Brush texture in a message box:

```
Sub SelectTexture()
With ActiveShape.Fill.Texture
    .Select "Air Brush", "Samples"
    MsgBox .GetPropertiesCount
End With
End Sub
```

{button ,AL(`CLS_TextureFill;FNC_Select')} **Related Topics**

TextureFill.SetProperties

Sub **SetProperties**(ByRef **SettingArray**() AS Variant)

TextureFill

Description

The **SetProperties** method allows you to set all the properties of an texture fill object in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

In order to use the **SetProperties** method, you must pass the following parameters:

Parameters	Description
SettingArray	Specifies the **TextureFillProperty** to set with the **SetProperties** method.

{button ,AL(`CLS_TextureFill;FNC_SetProperties')} **Related Topics**

cdrTexturePropertyNUM=0

cdrTexturePropertyRGB=1

cdrTexturePropertyHSB=2

cdrTexturePropertyCMYK=3

TextureFillProperty properties

TextureFillProperty Legend

- Name
- Type
- Value

TextureFillProperty

Class **TextureFillProperty**

Properties Referenced By

The **TextureFillProperty** class defines the properties of texture fill objects and describes the look and behavior of the properties through its properties and methods. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Texture fills, unlike tiled bitmap fills, fill a designated area with one image instead of with a series of repeating images. Texture fills increase the size of a file and the time it takes to print. Therefore, you may want to use these fills sparingly, especially with larger objects.

You can use colors from any color model or palette for texture fills. Since texture fills can only hold RGB colors, however, this can cause a color shift when you display or print the files.

Corel TEXTURE lets you design bitmap texture fills and modify preset textures. You can recreate the natural textures of wood, clouds, stone, ripples, waves, and wrinkles, or create artificial patterns such as checkers, dots, lines, and swirls. Corel TEXTURE lets you control lighting, design, color combinations, and gradations. The Texture wizard guides you through the process, or you can start with a blank texture.

{button ,AL(`CLS_TextureFillProperty')} **Related Topics**

TextureFillProperty.Name

Property **Name** As String

Description

The **Name** returns the name of a texture fill property in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

This property returns a Read-Only value.

Example

The following code example displays the name of the first property in the **TextureFillProperties** collection in CorelDRAW:

```
Sub Property()
MsgBox ActiveShape.Fill.Texture.Properties(1).Name
End Sub
```

{button ,AL(`CLS_TextureFillProperty;FNC_Name')} **Related Topics**

TextureFillProperty.Type

Property **Type** As **cdrTexturePropertyType**

Description

The **Type** returns the property type of a texture fill property in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

This property returns a Read-Only value of cdrTexturePropertyType.

Example

The following code example displays the property type of the first property in the **TextureFillProperties** collection in CorelDRAW:

```
Sub Property()
MsgBox ActiveShape.Fill.Texture.Properties(1).Type
End Sub
```

{button ,AL(`CLS_TextureFillProperty;FNC_Type')} **Related Topics**

TextureFillProperty.Value

Property **Value** As Variant

Description

The **Value** property returns or sets the value of a texture fill property in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Example

The following code example displays the value of the first property in the **TextureFillProperties** collection in CorelDRAW:

```
Sub Property()
MsgBox ActiveShape.Fill.Texture.Properties(1).Value
End Sub
```

{button ,AL(`CLS_TextureFillProperty;FNC_Value')} **Related Topics**

TextureFillProperties properties

TextureFillProperties Legend

- Count

- Item

TextureFillProperties

Class **TextureFillProperties**

Properties Referenced By

The **TextureFillProperties** class defines the a collection of properties of texture fill cobjects and describes the look and behavior of the collection through its properties and methods. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Texture fills, unlike tiled bitmap fills, fill a designated area with one image instead of with a series of repeating images. Texture fills increase the size of a file and the time it takes to print. Therefore, you may want to use these fills sparingly, especially with larger objects.

You can use colors from any color model or palette for texture fills. Since texture fills can only hold RGB colors, however, this can cause a color shift when you display or print the files.

Corel TEXTURE lets you design bitmap texture fills and modify preset textures. You can recreate the natural textures of wood, clouds, stone, ripples, waves, and wrinkles, or create artificial patterns such as checkers, dots, lines, and swirls. Corel TEXTURE lets you control lighting, design, color combinations, and gradations. The Texture wizard guides you through the process, or you can start with a blank texture.

{button ,AL(`CLS_TextureFillProperties')} **Related Topics**

TextureFillProperties.Count

Property **Count** As Long

Description

The **Count** property returns the number of texture fill properties in the TextureFillProperties collection in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

This property returns a Read-Only value.

Example

The following code example counts the number of texture fill properties in the TextureFillProperties collection in CorelDRAW:

```
Sub Properties()
MsgBox ActiveShape.Fill.Texture.Properties.Count
End Sub
```

{button ,AL(`CLS_TextureFillProperties;FNC_Count')} **Related Topics**

TextureFillProperties.Item

Property **Item**(ByVal **IndexOrName** As Variant) As **TextureFillProperty**

Description

The **Item** property returns a value associated with a specific property in the TextureFillProperties collection in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

This property returns a Read-Only value.

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a TextureFillProperties collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each tesxture fill property.

Example

The following code example displays the name of the first property in the **TextureFillProperties** collection in CorelDRAW:

```
Sub Property()
MsgBox ActiveShape.Fill.Texture.Properties(1).Name
End Sub
```

{button ,AL(`CLS_TextureFillProperties;FNC_Item')} **Related Topics**

PostScriptFill properties

PostScriptFill Legend

- Index

- Name
  Properties

PostScriptFill methods

PostScriptFill Legend

- Select
- SetProperties

PostScriptFill

Class **PostScriptFill**

Properties Methods Referenced by

The **PostScriptFill** class defines the characteristics of postscript fill objects and describes the look and behavior of the objects through its properties and methods. Postscript fills are types of pattern fill designed using the PostScript language.

Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view. You can enable or disable the Show PostScript fills to display the fill.

PostScript is a page description language that sends printing instructions to a PostScript device. All the elements in a print job (for example, curves and text) are represented by lines of PostScript code that the printing device uses to produce the document.

PostScript is not the only method for sending printing instructions, and some printing devices are not compatible with PostScript. However, there are several functions that are unavailable when you do not use PostScript. For example, you cannot adjust halftone screens.

There are three levels of PostScript. PostScript 1, the first PostScript language, has certain limitations. PostScript 2 greatly reduces potential printing errors. PostScript 3, the latest version of PostScript, is faster than previous versions, which have been largely eliminated in PostScript 2 or PostScript 3.

The settings listed in the Parameters section of the PostScript Texture dialog box vary depending on the type of PostScript texture fill you choose.

{button ,AL(`CLS_PostScriptFill')} **Related Topics**

PostScriptFill.Name

Property **Name** AS String

PostScriptFill

Description

The **Name** property returns the name of a postscript fill object in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language.

Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view. You can enable or disable the Show PostScript fills to display the fill.

The **Name** property returns a Read-Only value.

Example

The following code example applies the "Cracks" postscript fill to the active shape using the **Select** method and sets the properties of the fill with the **SetProperties** method. 15 is the number of cracks, 12 is maximum crack length, 5 is the minimum crack length, 30 is the Step length (space between cracks). A message and displays the name and index number of the postscript fill object:

```
Sub PostFillName()
With ActiveShape.Fill.PostScript
    .Select ("Cracks")
    .SetProperties 15, 12, 5, 30      'distance measured in document units
    MsgBox .Index
    MsgBox .Name
    MsgBox .Properties(1)
End With
End Sub
```

{button ,AL(`CLS_PostScriptFill;FNC_Name')} **Related Topics**

PostScriptFill.Index

Property **Index** AS Long

PostScriptFill

Description

The **Index** property returns the index number associated with a postscript fill object in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language. The **Index** property uniquely identifies a postscript fill object.

Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view. You can enable or disable the Show PostScript fills to display the fill.

The **Index** property returns a Read-Only value.

Example

The following code example applies the "Cracks" postscript fill to the active shape using the **Select** method and sets the properties of the fill with the **SetProperties** method. 15 is the number of cracks, 12 is maximum crack length, 5 is the minimum crack length, 30 is the Step length (space between cracks). A message and displays the name and index number of the postscript fill object:

```
Sub PostFillName()
With ActiveShape.Fill.PostScript
    .Select ("Cracks")
    .SetProperties 15, 12, 5, 30      'distance measured in document units
    MsgBox .Index
    MsgBox .Name
    MsgBox .Properties(1)
End With
End Sub
```

{button ,AL(`CLS_PostScriptFill;FNC_Index')} **Related Topics**

PostScriptFill.Properties

Property **Properties**(ByVal **Index** AS Long) AS Long

PostScriptFill

Description

The **Properties** property returns or sets a specified property of a postscript fill by referencing the index number of a property. Postscript fills are types of pattern fill designed using the PostScript language.

Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view. You can enable or disable the Show PostScript fills to display the fill.

In order to reference the **Properties** property, you must pass the **Index** value of a postscript fill as a parameter:

Parameters	Description
Index	The **Index** parameter specifies a property of a postscript fill object in CorelDRAW. For example, if a postscript fill has 4 properties and an Index value of 2 is passed as the parameter, a value associated with the second property of the fill is returned. For example, The "Bricks" postscript fill has four properties. `PostScriptFill.Properties(2)` returns the line width of the fill.

Example

The following code example applies the "Cracks" postscript fill to the active shape using the **Select** method and sets the properties of the fill with the **SetProperties** method. 15 is the number of cracks, 12 is maximum crack length, 5 is the minimum crack length, 30 is the Step length (space between cracks). A message and displays the name and index number of the postscript fill object:

```
Sub PostFillName()
With ActiveShape.Fill.PostScript
    .Select ("Cracks")
    .SetProperties 15, 12, 5, 30      'distance measured in document units
    MsgBox .Index
    MsgBox .Name
    MsgBox .Properties(1)
End With
End Sub
```

{button ,AL(`CLS_PostScriptFill;FNC_Properties')} **Related Topics**

PostScriptFill.Select

Sub **Select**(ByVal **IndexOrName** AS Variant)

PostScriptFill

Description

The **Select** method chooses a specified postscript fill object by referencing the fill's name or index number in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language.

Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view. You can enable or disable the Show PostScript fills to display the fill.

When using the Select method, you must pass the **IndexorName** value of a postscript fill as a parameter:

Parameters	Description
IndexOrName	The **Index** parameter specifies the index number of a postscript fill object in CorelDRAW. The **Index** parameter uniquely identifies a postscript fill. The name parameter is the string value that uniquely identifies the postscript fill.

Example

The following code example applies the "Cracks" postscript fill to the active shape using the **Select** method and sets the properties of the fill with the **SetProperties** method. 15 is the number of cracks, 12 is maximum crack length, 5 is the minimum crack length, 30 is the Step length (space between cracks). A message and displays the name and index number of the postscript fill object:

```
Sub PostFillName()
With ActiveShape.Fill.PostScript
    .Select ("Cracks")
    .SetProperties 15, 12, 5, 30      'distance measured in document units
    MsgBox .Index
    MsgBox .Name
    MsgBox .Properties(1)
End With
End Sub
```

{button ,AL(`CLS_PostScriptFill;FNC_Select')} **Related Topics**

PostScriptFill.SetProperties

Sub **SetProperties**(ByVal **Param1** AS Long, ByVal Param2 AS Long, ByVal Param3 AS Long, ByVal Param4 AS Long, ByVal Param5 AS Long)

PostScriptFill

Description

The **SetProperties** method allows you to set all properties of a postscript fill object in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language. Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture, unless you are in Enhanced view.

The **SetProperties** method has placeholders for a range of postscript fill properties. The maximum number of properties for any postscript fill object is 5. Properties may differ within each postscript fill object.

In order to use the **SetProperties** method, you must pass several property values as parameters:

Parameters	Description
Param1	The **Param1** parameter references the first property value of a postscript fill object. For example, the first property of the "Bars" postscript fill is "Width".
Param2	The **Param2** parameter references the second property value of a postscript fill object. This value is optional and the default value is 0. For example, the **Param2** value of the "Bars" fill references the "Spacing (%)" of the fill.
Param3	The **Param3** parameter references the third property value of a postscript fill object. This value is optional and the default value is 0. For example, the **Param3** value of the "Bars" fill references the "Maximum gray" value of the fill.
Param4	The **Param4** parameter references the fourth property value of a postscript fill object. This value is optional and the default value is 0. For example, the **Param4** value of the "Bars" fill references the "Minimum gray" value of the fill.
Param5	The **Param5** parameter references the fifth property value of a postscript fill object. This value is optional and the default value is 0. For example, the **Param5** value of the "Checks" fill references the "Line width" value of the fill.

Example

The following code example applies the "Cracks" postscript fill to the active shape and sets the properties of the fill with the **SetProperties** method. 15 is the number of cracks, 12 is maximum crack length, 5 is the minimum crack length, 30 is the Step length (space between cracks). A message box displays the number of cracks:

```
Sub PostFillName()
With ActiveShape.Fill.PostScript
    .Select ("Cracks")
    .SetProperties 15, 12, 5, 30      'distance measured in document units
    MsgBox .Properties(1)
End With
End Sub
```

{button ,AL(`CLS_PostScriptFill;FNC_SetProperties')} **Related Topics**

Fill properties

Fill Legend

- Fountain
- Pattern
- PostScript
- Texture
- Type
- UniformColor

Fill methods

Fill Legend

- ApplyFountainFill
- ApplyNoFill
- ApplyPatternFill
- ApplyPostscriptFill
- ApplyTextureFill
- ApplyUniformFill

Fill

Class **Fill**

Properties Methods Referenced by

The **Fill** class defines the characteristics of fill objects and describes the look and behavior of the objects through its properties and methods. Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

In CorelDRAW, fills can be applied to any drawn object or curve. In Corel PHOTO-PAINT, fills can be applied to the contents of rectangles, polygons, etc., but are more often applied to portions of your bitmap image using the Fill tool.

The Scrapbook's Favorite Fills And Outlines page offers access to preset fills and outlines provided by CorelDRAW, as well as fills or outlines that you've created and stored. You can apply these fills and outlines to any object you create in CorelDRAW. Fills appear in closed shapes only.

CorelDRAW lets you copy and paste fill and outline colors from one object to another using the Eyedropper and Paintbucket tools.

{button ,AL(`CLS_Fill')} **Related Topics**

Fill.Type

Property **Type** AS **cdrFillType**

Fill

Description

The **Type** property returns a value associated with the fill type an object in CorelDRAW. Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Note that in order to change the fill type you should use the corresponding "Apply"method of the Fill object. The exception is applying a uniform fill. Assigning a color to **UniformColor** property will automatically set the fill type to cdrUniformFill.

The **Type** property returns a Read-Only value of cdrFillType.

Example

The following code example set the fill type in the active shape of CorelDRAW. The fill is a fountain fill:

```
Sub FillType()
With ActiveShape.Fill
.Type = cdrFountainFill
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_Type')} **Related Topics**

cdrNoFill=0

cdrUniformFill=1

cdrFountainFill=2

cdrPostscriptFill=3

cdrMonoBitmapFill=4

cdrReservedFill=5

cdrColorBitmapFill=6

cdrVectorFill=7

cdrTextureFill=8

cdrPatternFill=9

Fill.UniformColor

Property **UniformColor** AS **Color**

Fill

Description

The **UniformColor** property returns or sets a uniform color in a fill object in CorelDRAW. Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

A Uniform colors palette is an independent palette (not based on a color-matching system or your image) that provides 256 colors that are uniformly spread between red, green, and blue.

Assigning a color to **UniformColor** property will automatically set the fill type to cdrUniformFill.

Example

The following code example applies a uniform color to the active shape in CorelDRAW. The orange color is created based on the RGB color model:

```
Sub FillColor()
With ActiveShape.Fill.UniformColor
    .RGBAssign 200, 100, 23
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_UniformColor')} **Related Topics**

Fill.Fountain

Property **Fountain** AS **FountainFill**

Fill

Description

The **Fountain** property returns or sets the properties of a fountain fill in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Example

The following code example displays the palette ID number of the end color used in the fountain fill effect in the active shape of CorelDRAW:

```
Sub FillFountain()
With ActiveShape.Fill.Fountain
    MsgBox .EndColor.PaletteID
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_Fountain')} **Related Topics**

Fill.Pattern

Property **Pattern** AS **PatternFill**

Fill

Description

The **Pattern** property returns or sets the properties of a pattern fill in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Example

The following code example sets the pattern fill type used in the pattern fill of the active shape in CorelDRAW. The pattern fill is a two-color pattern fill:

```
Sub FillPattern()
With ActiveShape.Fill.Pattern
    .Type = cdrTwoColorPattern
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_Pattern')} **Related Topics**

Fill.Texture

Property **Texture** AS **TextureFill**

Fill

Description

The **Texture** property returns or sets the properties of a texture fill in CorelDRAW. A texture fill is a fractally generated fill, such as water, minerals, and clouds, that you can use to give your objects a natural appearance. Texture fills, unlike tiled bitmap fills, fill a designated area with one image instead of with a series of repeating images.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Example

The following code example displays the style name of the texture fill used in the active shape of CorelDRAW:

```
Sub FillTexture()
With ActiveShape.Fill.Texture
    MsgBox .StyleName
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_Texture')} **Related Topics**

Fill.PostScript

Property **PostScript** AS **PostScriptFill**

Fill

Description

The **PostScript** property returns or sets the properties of a postscript fill in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language. Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Example

The following code example displays the name of the postscript fill used in the active shape of CorelDRAW:

```
Sub PostscriptName()
With ActiveShape.Fill. PostScript
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_PostScript')} **Related Topics**

Fill.ApplyNoFill

Sub **ApplyNoFill**()

Fill

Description

The **ApplyNoFill** method ensures that no fill is added to a shape object in CorelDRAW. If a fill is currently applied to an object, the **ApplyNoFill** method removes the fill.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

Example

The following code example applies no fill to the active shape in CorelDRAW:

```
Sub FillNothing()
With ActiveShape.Fill
    .ApplyNoFill
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyNoFill')} **Related Topics**

Fill.ApplyUniformFill

Sub **ApplyUniformFill**(ByRef **Color** AS **Color**)

Fill

Description

The **ApplyUniformFill** method applies a uniform fill color to a shape object in CorelDRAW. A Uniform colors palette is an independent palette (not based on a color-matching system or your image) that provides 256 colors that are uniformly spread between red, green, and blue.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

In order to apply the **ApplyUniformFill** method, you must pass the **Color** value as a parameter:

Parameters	Description
Color	The **Color** parameter defines an existing or new color that is applied as a uniform color to a selected shape object. It uses an existing color component in a palette or a **CreateColor** method (i.e. **CreateCMYKColor**, **CreateRGBColor**) to specify a color. A color in a palette is identified by referencing the palette and the index of the color in the palettet. For example, `ActivePalette.Colors(50)`, references the fiftieth color in the active palette of CorelDRAW.

Example

The following code examples apply a uniform fill to the active shape object in CorelDRAW. The first example adds a uniform color from the active palette, while the second example adds a newly created color as a uniform fill effect:

```
Sub UniformPalette()
Dim PaletteColor as Color
Set PaletteColor = ActivePalette.Colors(10)
'applies the tenth color in the current color palette
ActiveShape.Fill.ApplyUniformFill PaletteColor
End Sub


Sub UniformNew()
Dim NewColor as Color
Set NewColor = CreateColorEx (5, 20, 134, 23)
'5 references the RGB color model - the new color is dark green
ActiveShape.Fill.ApplyUniformFill NewColor
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyUniformFill')} **Related Topics**

Fill.ApplyFountainFill

Function **ApplyFountainFill**(ByRef StartColor AS **Color**, ByRef EndColor AS **Color**, ByVal Type AS **cdrFountainFillType**, ByVal Angle AS Double, ByVal Steps AS Long, ByVal EdgePad AS Long, ByVal MidPoint AS Long, ByVal BlendType AS **cdrFountainFillBlendType**) AS **FountainFill**

Fill

Description

The **ApplyFountainFill** method applies a fountain fill effect to a shape object in CorelDRAW. A fountain fill is a complex fill that displays a progression between two colors that follow a linear, radial, conical, or square path. Fountain fills are also known as gradient or graduated fills. You can create a direct blend from one color to another or a cascade of different colors. You can also use preset fountain fills to create neon tubes, metal cylinders, and other effects.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

In order to apply the **ApplyFountainFill** method, you must pass the several values as parameters:

Parameters	Description
StartColor	The **StartColor** parameter defines the color that appears at the beginning of a fountain fill effect. The value is optional and the default value is Nothing (0).
EndColor	The **EndColor** parameter defines the color that appears at the end of a fountain fill effect. The value is optional and the default value is Nothing (0).
Type	The **Type** parameter defines the fountain fill type in the fill effect. A fountain fill can be linear, conical, radial, or square. The **Type** parameter returns cdrFountainFillType. This value is optional and the default value is cdrLinearFountainFill (1).
Angle	The **Angle** parameter defines the degree of the angle in linear, conical or square fountain fill effects. Changing the angle of gradation affects the slant of the fountain fill. Radial fountain fills progress in a series of concentric circles, so you can't change their angle. Positive values rotate the fill counterclockwise; negative values rotate it clockwise. This value is optional and the default value is 0.
Steps	The **Steps** parameter identifies the number of bands (steps) used to display a fountain fill. When you create a fountain fill, the space required to blend the colors is divided by the number of fountain steps displayed in the Steps box. This value is optional and the default value is 0.
EdgePad	The **EdgePad** parameter defines the length of solid colors at the beginning and end of the fountain fill before the start blending with the next color in the fountain fill.You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors. This value is optional and the default value is 0.
MidPoint	The **MidPoint** is an imaginary line between two colors in a fountain fill. The value of the mid-point represents its position in relation to two fountain fill colors. By changing this value, you can set the point at which two colors in a fountain fill converge. For example, in a two-color fountain fill using the colors black and white, a value of 50 positions the mid-point in the center of the fill so that half of the fill is black and half is white. Increasing the mid-point value to 99 results in a fountain fill dominated by black. Decreasing the mid-point value to 1 results in a fountain fill dominated by white. This value is optional and the default value is 50.
Blend	The **Blend** parameter determines the blend type in the fountain fill effect. The blend type defines how colors will appear with other colors in the fill effect. This value returns cdrFountainFillBlendType. The value is optional and the default value is cdrDirectFountainFillBlend

(0).

Example

The following code example selects an object in the active document and determines if the object is a curve object. If the selection is a curve, it changes its Closed status to True, joining all the segments in the shape to make a self-contained object. A color fountain fill is applied to the new closed curve object using the **CreateColorEx** method:

```
Sub CurveClosed()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    s.Curve.Closed = True
End If
s.Fill.ApplyFountainFill CreateColorEx(5005, 255, 0, 0), _
CreateColorEx(5005, 0, 0, 0), 1
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyFountainFill')} **Related Topics**

cdrDirectFountainFillBlend=0
cdrRainbowCWFountainFillBlend=1
cdrRainbowCCWFountainFillBlend=2
cdrCustomFountainFillBlend=3

Fill.ApplyPatternFill

Function **ApplyPatternFill**(ByVal **Type** AS **cdrPatternFillType**, ByVal FileName AS String, ByVal PatternCanvasIndex AS Long, ByRef FrontColor AS **Color**, ByRef EndColor AS **Color**, ByVal TransformWithShape AS Boolean) AS **PatternFill**

Fill

Description

The **ApplyPatternFill** method applies a pattern fill to a shape object in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

In order to use the **ApplyPatternFill** method, you must pass several values as parameters:

Parameters	Description
Type	The **Type** parameter identifies the type of pattern fill and returns a value of cdrPatternFillType.
FileName	The **FileName** parameter identifies the full path name of the graphic that is imported into CorelDRAW to be used as a pattern fill. A file name contains the computer's path and the name of the graphic file. This value is optional.
PatternCanvasIndex	The **PatternCanvasIndex** parameter identifies the pattern canvas in a pattern fill effect. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. Each canvas has an index number that uniquely identifies the canvas. This value is optional.
FrontColor	The **FrontColor** parameter identifies the foreground color in a pattern fill. This value is optional and the default value is Nothing(0).
EndColor	The **EndColor** parameter identifies the background color in a pattern fill. This value is optional and the default value is Nothing(0).
TransformWithShape	The **TransformWithShape** parameter is a True or False value, indicating if the pattern fill changes to fit its shape when the shape object is altered. If the value is True, the pattern fill will change according to changes in its shape object. This value is optional and the default value is False.

Example

The following code example applies a two color "checkers" pattern fill to the active shape in CorelDRAW. The fill will transform its shape when the object's shape is altered:

```
Sub FillPattern()
Dim ColorFront as Color
Dim ColorBack as Color
Set ColorFront = ActivePalette.Colors(3)
Set ColorBack = ActivePalette.Colors(4)
With ActiveShape.Fill
    .ApplyPatternFill cdrTwoColorPattern, "checkers", 1, ColorFront, ColorBack, True
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyPatternFill')} **Related Topics**

Fill.ApplyTextureFill

Function **ApplyTextureFill**(ByVal **TextureName** AS String, ByVal LibraryName AS String) AS **TextureFill**

Fill

Description

The **ApplyTextureFill** method applies a texture fill to a shape object in CorelDRAW. A texture fill is a random, fractally-generated fill that you can use to give your objects a natural appearance. Texture fills are fills that look like clouds, water, gravel, minerals, and other natural and artificial substances. CorelDRAW provides preset textures, and each texture has a set of options that you can change.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

In order to use the **ApplyTextureFill** method, you must pass the **TextureName** and **LibraryName** values as parameters:

Parameters	Description
TextureName	The **TextureName** property returns a string value associated with the name of a texture fill pattern in CorelDRAW. For example, "Air Brush" is texture fill in the "Samples" library of CorelDRAW.
LibraryName	The **LibraryName** parameter returns a string value associated with the name of a texture fill pattern library in CorelDRAW. A texture library is a collection of texture fill pattern files in CorelDRAW. There are several texture libraries included in CorelDRAW. For example, "Samples" is a texture library. The **LibraryName** parameter is optional.

Example

The following code example applies a texture fill pattern to the active shape in CorelDRAW. The texture pattern is called "Air Brush" and it is found in the "Samples" texture library:

```
Sub FillTexture()
With ActiveShape.Fill
    .ApplyTextureFill ("Air Brush", "Samples")
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyTextureFill')} **Related Topics**

Fill.ApplyPostscriptFill

Function **ApplyPostscriptFill**(ByVal **IndexOrName** AS Variant) AS **PostScriptFill**

Fill

Description

The **ApplyPostscriptFill** method applies a postscript fill effect to a shape object in CorelDRAW. Postscript fills are types of pattern fill designed using the PostScript language. Some textures are extremely complicated and require several minutes or more to print or update on the screen. Therefore, PostScript fills are displayed as the letters, PS, rather than as the texture.

Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image. You can change the appearance of an object by using a fill. To fill an object, you apply colors or patterns inside its borders. If you leave an object without a fill or remove its fill, the object is transparent. CorelDRAW includes Uniform fills, Fountain fills, Pattern fills, Texture fills, and PostScript fills. You can apply fills to specific objects or set defaults so that every object you draw has the same fill. You can also customize Fountain fills, Pattern fills, and Texture fills. You can also apply a mesh fill to an object.

In order to use the **ApplyPostscriptFill** method, you must pass the **IndexOrName** value as a parameter:

Parameters	Description
IndexOrName	The **Index** parameter specifies the index number of a postscript fill object in CorelDRAW. The **Index** parameter uniquely identifies a postscript fill. The name parameter is the string value that uniquely identifies the postscript fill.

Example

The following code example applies the predefined postscript fill "Cracks" to the active shape object in CorelDRAW:

```
Sub ApplyPostscript()
With ActiveShape.Fill
    .ApplyPostscriptFill ("Cracks")
End With
End Sub
```

{button ,AL(`CLS_Fill;FNC_ApplyPostscriptFill')} **Related Topics**

CorelScriptFile properties

CorelScriptFile Legend

- Application

- FileName
  Name

- Parent

CorelScriptFile methods

CorelScriptFile Legend

- Delete
- Play
- Translate

CorelScriptFile

Class **CorelScriptFile**

Properties Methods Referenced by

The **CorelScriptFile** class defines the characteristics of CorelSCRIPT file objects and describes the look and behavior of the objects through its properties and methods. Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. If you run a script for an application that is not running, Corel SCRIPT automatically starts the application.

A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

The Corel SCRIPT programming language is based on the BASIC programming language. If you're already familiar with a version of BASIC, you'll find the Corel SCRIPT programming language easy to read and understand.

For more information on Corel SCRIPT and its components, please view the Help File in the Corel SCRIPT Editor.

{button ,AL(`CLS_CorelScriptFile')} **Related Topics**

CorelScriptFile.Application

Property **Application** AS Object

CorelScriptFile

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW 10.0 in a message box:

```
Sub ScriptApplication()
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
       MsgBox "You are using CorelDRAW " & Version
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_Application')} **Related Topics**

CorelScriptFile.Parent

Property **Parent** AS Object

CorelScriptFile

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the caption of the active window in a message box:

```
Sub CorelScriptFileParent()
Dim csf As CorelScriptFile
With csf
    MsgBox .Parent.ActiveWindow.Caption
End With
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_Parent')} **Related Topics**

CorelScriptFile.Play

Sub **Play**(ByVal Document AS Object)

CorelScriptFile

Description

The **Play** method runs a Corel SCRIPT file in the active document of CorelDRAW. Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

In order to use the **Play** method, you must pass the **Document** value as a parameter:

Parameters	Description
Document	The **Document** parameter specifies which document will run the script file and activates that document prior to running the script. If a document is not specified with the Play method, the active document with run the script.

Example

The following code example creates an ellipse and runs a Corel SCRIPT file that applies a fountain fill to the ellipse:

```
Sub ScriptPlay()
Dim s As New Shape
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
Set s = ActiveLayer.CreateEllipse(4, 2, 1, 0)
csf.Play
'****csf.Play Documents (2) will run the script in the second open document
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_Play')} **Related Topics**

CorelScriptFile.Translate

Sub **Translate**(ByVal Document AS Object)

CorelScriptFile

Description

The **Translate** method converts a Corel SCRIPT file into a VBA code module in CorelDRAW. Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

A VBA code module is a programming package of Visual Basic for Applications code segments. VBA is a subset of the Microsoft Visual Basic (VB) object-oriented programming environment. VBA uses the Visual Basic Editor interactive development environment and the VB programming language to enhance applications by manipulating the application's objects, exposed by its object model. VBA can access other applications from within CorelDRAW 10.0 by referencing that application's object model components.

In order to use the **Translate** method, you must pass the **Document** value as a parameter:

Parameters	Description
Document	The **Document** parameter specifies which document will contain the generated VBA code module. If a document is not specified with the Translate method, the generated VBA code module will be stored in CorelDRAW 10.0 Global Macros.

Example

The following code example opens a fountain fill Corel SCRIPT file and translates it into a VBA code module, which is stored in the active document:

```
Sub ScriptTranslate()
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
csf.Translate ActiveDocument
'****csf.Translate will store the generated code in Global Macros
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_Translate')} **Related Topics**

CorelScriptFile.Name

Property **Name** AS String

CorelScriptFile

Description

The **Name** property returns or sets a string value that identifies a new VBA code module, when it is translated from a Corel SCRIPT file. Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

A VBA code module is a programming package of Visual Basic for Applications code segments. VBA is a subset of the Microsoft Visual Basic (VB) object-oriented programming environment. VBA uses the Visual Basic Editor interactive development environment and the VB programming language to enhance applications by manipulating the application's objects, exposed by its object model. VBA can access other applications from within CorelDRAW 10.0 by referencing that application's object model components.

By default, the module name is set to the original SCRIPT name, without the script extension (.csc).

Example

The following code example opens the a SCRIPT file and translates the file into a global macro VBA code module, giving it a specified name:

```
Sub ScriptName()
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
csf.Name = "AlienSkyModule"
csf.Translate
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_Name')} **Related Topics**

CorelScriptFile.FileName

Property **FileName** AS String

CorelScriptFile

Description

The **FileName** property returns a string value that identifies the full computer location and file name of a Corel SCRIPT file in CorelDRAW. Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

The **FileName** property returns a Read-Only value.

Example

The following code example opens the a SCRIPT file, displays the path name of the file in a message box, then creates an ellipse and applies a preset fountain fill:

```
Sub NameFile()
Dim s As New Shape
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
       MsgBox csf.FileName
Set s = ActiveLayer.CreateEllipse(4, 2, 1, 0)
csf.Play
End Sub
```

{button ,AL(`CLS_CorelScriptFile;FNC_FileName')} **Related Topics**

CorelScriptFile.Delete

Sub **Delete**()

CorelScriptFile

Description

The **Delete** method removes the original Corel SCRIPT file from CorelDRAW.Corel SCRIPT files are individual packages of Corel SCRIPT programming procedures that perform a specific task when used within the CorelDRAW application.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

Example

The following code example translates a script file into a VBA code module and deletes the original Corel SCRIPT file:

```
Sub ScriptDelete()
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Scripts\Fills\Fountain\aliensky.csc")
csf.Name = "AlienskyModule"
csf.Translate
csf.Delete
```

{button ,AL(`CLS_CorelScriptFile;FNC_Delete')} **Related Topics**

Effect properties

Effect Legend

- Application
- Blend
- CloneParent
- Clones
- Contour
- ControlPath
- Distortion
- DropShadow
- Envelope
- Extrude
- Lens
- Parent
- Perspective
- TextOnPath
- Type

Effect methods

Effect Legend

- Clear
- Separate

Effect

Class **Effect**

Properties Methods Referenced by

The **Effect** class defines the characteristics of effect objects and describes the look and behavior of the objects through its properties and methods.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

By blending two objects, you create a series of intermediate objects that show a transition in color, shape, and size. Three distortion effects let you change the shape of objects: Push and Pull, Zipper, and Twister. You can also change the shape of an object by applying an envelope to it then shaping the envelope. The object conforms to the shape of the envelope. By extruding an object, CorelDRAW adds surfaces to make it appear three-dimensional. Adding one and two-point perspective lets you create the illusion of depth and distance. You can create the illusion of depth between objects by adding drop shadows. You can apply transparencies, which are grayscale masks, to objects. Transparencies are applied on top of the object's current fill, therefore the object's color show through the transparency. The lenses in CorelDRAW let you change the appearance of objects you view through them. You can contour an object, creating the effect created by contour lines on a topographical map. When you create PowerClip objects, you place an object inside another object. One object becomes the contents; the other object becomes the container.

You can apply effects to both Paragraph text and Artistic text. However, some effects are exclusive to Artistic text and others to Paragraph text because CorelDRAW treats the two text types differently. Effects you can apply to Paragraph text frames include applying envelopes, drop shadows, and PowerClip objects. Effects you can apply to Artistic text include extrusions, blends, contours, distortions, envelopes, lenses, PowerClip objects, perspectives, and drop shadows.

{button ,AL(`CLS_Effect')} **Related Topics**

Effect.Application

Property **Application** AS **Application**

Effect

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Application')} **Related Topics**

Effect.Parent

Property **Parent** AS **Effects**

Effect

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Parent')} **Related Topics**

Effect.Type

Property **Type** AS **cdrEffectType**

Effect

Description

The **Type** property returns a value associated with an effect type in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

The **Type** property returns a value of cdrEffectType.

The **Type** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Type')} **Related Topics**

Effect.Blend

Property **Blend** AS **EffectBlend**

Effect

Description

The **Blend** property returns a value associated with a blend effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

The **Blend** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Blend')} **Related Topics**

Effect.ControlPath

Property **ControlPath** AS **EffectControlPath**

Effect

Description

The **ControlPath** property returns a value associated with the control path of an effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A control path is a curve that determines the shape of an Artistic Media stroke. To change the shape of the curve, you must edit the control path.

The **ControlPath** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_ControlPath')} **Related Topics**

Effect.Extrude

Property **Extrude** AS **EffectExtrude**

Effect

Description

The **Extrude** property returns a value associated with an extrude effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

An extrude effect is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

The **Extrude** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Extrude')} **Related Topics**

Effect.Envelope

Property **Envelope** AS **EffectEnvelope**

Effect

Description

The **Envelope** property returns a value associated with an envelope effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object.

The **Envelope** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Envelope')} **Related Topics**

Effect.TextOnPath

Property **TextOnPath** AS **EffectTextOnPath**

Effect

Description

The **TextOnPath** property returns a value associated with an text on path effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

Fitting text to a path aligns text objects along the path of a specified shape in CorelDRAW. The text follows the path of the shape object.

The **TextOnPath** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_TextOnPath')} **Related Topics**

Effect.DropShadow

Property **DropShadow** AS **EffectDropShadow**

Effect

Description

The **DropShadow** property returns a value associated with an drop shadow effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A drop shadow effect creates a dark area, in the shape of the object, around the object in CorelDRAW. It is a darker duplication of the object that is slightly offset from the object's original position.

The **DropShadow** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_DropShadow')} **Related Topics**

Effect.Contour

Property **Contour** AS **EffectContour**

Effect

Description

The **Contour** property returns a value associated with a contour effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A contour is a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

The spaces between contour lines are filled with colors that follow a progression from the original object to the last shape created. If there is a difference in color between the contour lines and the outline of the original object, a second progression occurs. You can modify both color progressions to get the look you want.

The **Contour** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Contour')} **Related Topics**

Effect.Distortion

Property **Distortion** AS **EffectDistortion**

Effect

Description

The **Distortion** property returns a value associated with a distortion effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A distortion effect is a lack of proportionality in an image that results from defects in the optical system.

The **Distortion** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Distortion')} **Related Topics**

Effect.Lens

Property **Lens** AS **EffectLens**

Effect

Description

The **Lens** property returns a value associated with a lens effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

The **Lens** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Lens')} **Related Topics**

Effect.Perspective

Property **Perspective** AS **EffectPerspective**

Effect

Description

The **Perspective** property returns a value associated with a perspective effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

Perspective is an effect created by lengthening or shortening sides of an object to create the impression that the object is receding from view in two directions. You can create two-point perspective by using the Add Perspective command in the Effects menu.

The **Perspective** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Perspective')} **Related Topics**

Effect.Clones

Property **Clones** AS **Effects**

Effect

Description

The **Clones** property returns a value associated with the collection of effects cloned from the current object in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

You can also clone a special effect that is applied to an object and apply it to other objects. Objects with a cloned effect take on all changes that are made to that effect in the master.

The **Clones** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Clones')} **Related Topics**

Effect.CloneParent

Property **CloneParent** AS **Effect**

Effect

Description

The CloneParent property returns a value associated with the parent object of the clone effects collection in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

In order for this property to be effective, there must be a clone-master relationship between objects in the active document and the clone object must be selected object in CorelDRAW.

The **CloneParent** property returns a Read-Only value.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_CloneParent')} **Related Topics**

Effect.Clear

Sub **Clear**()

Effect

Description

The **Clear** method removes an effect from an object in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Clear')} **Related Topics**

Effect.Separate

Function **Separate**() AS **ShapeRange**

Effect

Description

The **Separate** method separates an effect from its object. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

Example

Example of usage goes here

{button ,AL(`CLS_Effect;FNC_Separate')} **Related Topics**

Effects properties

Effects Legend

- Application
- Count
-
- Item
- Parent

Effects

Class **Effects**

Properties Referenced by

The **Effects** class defines the characteristics of **Effects** collection objects and describes the look and behavior of the collection objects through its properties and methods.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

By blending two objects, you create a series of intermediate objects that show a transition in color, shape, and size. Three distortion effects let you change the shape of objects: Push and Pull, Zipper, and Twister. You can also change the shape of an object by applying an envelope to it then shaping the envelope. The object conforms to the shape of the envelope. By extruding an object, CorelDRAW adds surfaces to make it appear three-dimensional. Adding one and two-point perspective lets you create the illusion of depth and distance. You can create the illusion of depth between objects by adding drop shadows. You can apply transparencies, which are grayscale masks, to objects. Transparencies are applied on top of the object's current fill, therefore the object's color show through the transparency. The lenses in CorelDRAW let you change the appearance of objects you view through them. You can contour an object, creating the effect created by contour lines on a topographical map. When you create PowerClip objects, you place an object inside another object. One object becomes the contents; the other object becomes the container.

You can apply effects to both Paragraph text and Artistic text. However, some effects are exclusive to Artistic text and others to Paragraph text because CorelDRAW treats the two text types differently. Effects you can apply to Paragraph text frames include applying envelopes, drop shadows, and PowerClip objects. Effects you can apply to Artistic text include extrusions, blends, contours, distortions, envelopes, lenses, PowerClip objects, perspectives, and drop shadows.

{button ,AL(`CLS_Effects')} **Related Topics**

Effects.Application

Property **Application** AS **Application**

Effects

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub EffectsApp()
With ActiveShape.Effects
    MsgBox "You are using CorelDRAW " & .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Effects;FNC_Application')} **Related Topics**

Effects.Parent

Property **Parent** AS **Shape**

Effects

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the type of the **Effects** collection's parent shape in a message box:

```
Sub EffectsParent()
With ActiveShape.Effects
    MsgBox .Parent.Type
End With
End Sub
```

{button ,AL(`CLS_Effects;FNC_Parent')} **Related Topics**

Effects.Item

Property **Item**(ByVal **Index** AS Long) AS **Effect**

Effects

Description

The **Item** property returns a value associated with the index number of an effect in the **Effects** collection of CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

`Effects(2)` refers to the second effect in the **Effects** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `Effects.Item(2)` is the same as `Effects(2)` - they both reference the second effect in the **Effects** collection.

You must reference an effect in the collection by passing its index number as a parameter:

Parameters	Description
Index	**Index** is a preset placeholder for each effect in an **Effects** collection; it uniquely identifies each member of the collection.

Example

The following code example displays the effect type associated with the first effect in the **Effects** collection that is applied to the current shape in CorelDRAW. An effect must be applied to the active shape in order for the type to display in a message box:

```
Sub EffectsItem()
With ActiveShape.Effects
    MsgBox .Item(1).Type
End With
End Sub
```

{button ,AL(`CLS_Effects;FNC_Item')} **Related Topics**

cdrBlend=0

cdrExtrude=1

cdrEnvelope=2

cdrTextOnPath=3

cdrControlPath=4

cdrDropShadow=5

cdrContour=6

cdrTransparency=7

cdrDistortion=8

cdrLens=9

cdrPerspective=10

cdrPowerClip=11

Effects.Count

Property **Count** AS Long

Effects

Description

The **Count** property returns the number of effects in the **Effects** collection of CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

The **Count** property returns a Read-Only value.

Example

The following code example displays the number of effects in the Effects collection that are applied to the active shape in CorelDRAW:

```
Sub EffectsCount()
With ActiveShape.Effects
    MsgBox .Count
End With
End Sub
```

{button ,AL(`CLS_Effects;FNC_Count')} **Related Topics**

EffectBlend properties

EffectBlend Legend

- AccelerateSize
- Angle
- Application

- BlendGroup
  - ColorAcceleration
  - ColorBlendType
  - EndShape
  - EndShapeOffset
  - EndShapePoint
  - FullPath
  - LinkAcceleration
  - Loop
  - MapNodes
  - Mode

- Parent
  - Path
  - RotateShapes
  - Spacing
  - SpacingAcceleration
  - StartShape
  - StartShapeOffset
  - StartShapePoint
  - Steps

EffectBlend methods

EffectBlend Legend

- CopyFrom
- FuseEnd
- FuseStart
- Split

EffectBlend

Class **EffectBlend**

Properties Methods Referenced by

The **EffectBlend** class defines the characteristics of blend effect objects and describes the look and behavior of the objects through its properties and methods.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a number of different blends, including straight-line blends and compound blends, among others. When you've created a blend, you change its intermediate objects. For example, you can set the number of steps and distance between intermediate objects. You can also change the entire blend. For example, you can change the start or end object, split the blend, or change the blend path.

CorelDRAW offers a variety of features for creating interesting blend effects, such as straight-line blends, blending along a path, compound blends, and copying and cloning blends.

Editing blends lets you customize their appearance. You can manipulate a blend's intermediate objects and change its path. You can also work with the individual components of a blend by separating it.

You can edit blends by selecting changing the start and end objects of a blend. By reversing a blend's direction, the blend progresses from the end object to the start object. You can also split blends to create a compound blend or fuse the components of a compound blend to create a single blend. When you split a blend, the object you choose to split the blend becomes the end object for one component in the blend and the start object for the other.

When you blend an object along a path, you can select the path and apply the blend to new path. You can also remove a blend from its path. When you remove a blend from its path, the start and end objects remain stationary and the intermediate objects revert to their original, straight-line path.

You can separate a blend into four possible components: the start object, the end object, the intermediate objects, and the path (if the objects were blended along a path). You can also clear a blend by removing the intermediate objects and leaving the start and end objects.

{button ,AL(`CLS_EffectBlend')} **Related Topics**

EffectBlend.Application

Property **Application** AS **Application**

EffectBlend

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectBlend;FNC_Application')} **Related Topics**

EffectBlend.Parent

Property **Parent** AS **Effect**

EffectBlend

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectBlend;FNC_Parent')} **Related Topics**

EffectBlend.StartShape

Property **StartShape** AS **Shape**

EffectBlend

Description

The **StartShape** property returns or set the starting shape in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

{button ,AL(`CLS_EffectBlend;FNC_StartShape')} **Related Topics**

EffectBlend.EndShape

Property **EndShape** AS **Shape**

EffectBlend

Description

The **EndShape** property returns or set the ending shape in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

{button ,AL(`CLS_EffectBlend;FNC_EndShape')} **Related Topics**

EffectBlend.BlendGroup

Property **BlendGroup** AS **Shape**

EffectBlend

Description

The **BlendGroup** property returns the blend group in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectBlend;FNC_BlendGroup')} **Related Topics**

EffectBlend.Path

Property **Path** AS **Shape**

EffectBlend

Description

The **Path** property returns or sets the blend path in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_Path')} **Related Topics**

EffectBlend.StartShapeOffset

Property **StartShapeOffset** AS Double

EffectBlend

Description

The **StartShapeOffset** returns or set the offset value of the blend starting shape in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_StartShapeOffset')} **Related Topics**

EffectBlend.EndShapeOffset

Property **EndShapeOffset** AS Double

EffectBlend

Description

The **EndShapeOffset** returns or set the offset value of the blend ending shape in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_EndShapeOffset')} **Related Topics**

cdrBlendSteps=0

cdrBlendSpacing=1

EffectBlend.Mode

Property **Mode** AS **cdrBlendMode**

EffectBlend

Description

The **Mode** property returns or set the blend mode of a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

This property returns cdrBlendMode.

{button ,AL(`CLS_EffectBlend;FNC_Mode')} **Related Topics**

EffectBlend.Steps

Property **Steps** AS Long

EffectBlend

Description

The **Steps** property returns or sets the number of steps in a blend effect in CorelDRAW. The number of steps in a contour effect is linked to the number of lines that appear in the effect.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_Steps')} **Related Topics**

EffectBlend.Spacing

Property **Spacing** AS Double

EffectBlend

Description

The **Spacing** property returns or set the blend spacing in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_Spacing')} **Related Topics**

EffectBlend.Angle

Property **Angle** AS Double

EffectBlend

Description

The **Angle** property returns or set the blend angle in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_Angle')} **Related Topics**

EffectBlend.Loop

Property **Loop** AS Boolean

EffectBlend

Description

The **Loop** property returns or sets a True or False value that indicates whether or not a blend loops in a blend effect in CorelDRAW. Looping allows you to rotate the blend halfway between the start and end objects' centers of rotation.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_Loop')} **Related Topics**

EffectBlend.FullPath

Property **FullPath** AS Boolean

EffectBlend

Description

The **FullPath** property returns or sets a True or False value that indicates whether or not a blend effect follows the full path of an object in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_FullPath')} **Related Topics**

EffectBlend.RotateShapes

Property **RotateShapes** AS Boolean

EffectBlend

Description

The **RotateShapes** property returns or sets a True or False value that indicates whether or not the blend effect rotates its shape to follow a path in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_RotateShapes')} **Related Topics**

EffectBlend.ColorBlendType

Property **ColorBlendType** AS **cdrFountainFillBlendType**

EffectBlend

Description

The **ColorBlendType** property returns or set the color blend type of a fountain fill blend type in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

This property returns cdrFountainFillBlendType.

{button ,AL(`CLS_EffectBlend;FNC_ColorBlendType')} **Related Topics**

EffectBlend.SpacingAcceleration

Property **SpacingAcceleration** AS Long

EffectBlend

Description

The **SpacingAcceleration** property returns or sets the rate of object acceleration in a blend effect in CorelDRAW. You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_SpacingAcceleration')} **Related Topics**

EffectBlend.ColorAcceleration

Property **ColorAcceleration** AS Long

EffectBlend

Description

The **ColorAcceleration** property returns or sets the rate of color acceleration in a blend effect in CorelDRAW. Higher numbers allow the colors to move quicker through the spectrum as they approach the end object.

You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_ColorAcceleration')} **Related Topics**

EffectBlend.LinkAcceleration

Property **LinkAcceleration** AS Boolean

EffectBlend

Description

The **LinkAcceleration** property returns or sets a True or False value that links color acceleration to object spacing acceleration in a blend effect in CorelDRAW. You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_LinkAcceleration')} **Related Topics**

EffectBlend.AccelerateSize

Property **AccelerateSize** AS Boolean

EffectBlend

Description

The **AccelerateSize** property returns or sets a True or False value that indicates whether or not the size of a blend accelerates along the path in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

{button ,AL(`CLS_EffectBlend;FNC_AccelerateSize')} **Related Topics**

EffectBlend.MapNodes

Property **MapNodes** AS Boolean

EffectBlend

Description

The **MapNodes** property returns or sets a True or False value that indicates whether or not the nodes are mapped in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

You can specify which nodes CorelDRAW treats as the first node on start and end object's first nodes. These nodes determine how the intermediate objects are transformed from the start object to become the end object.

{button ,AL(`CLS_EffectBlend;FNC_MapNodes')} **Related Topics**

EffectBlend.StartShapePoint

Property **StartShapePoint** AS **ShapePoint**

EffectBlend

Description

The **StartShapePoint** property returns or set the starting mapped node in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

You can specify which nodes CorelDRAW treats as the first node on start and end object's first nodes. These nodes determine how the intermediate objects are transformed from the start object to become the end object.

{button ,AL(`CLS_EffectBlend;FNC_StartShapePoint')} **Related Topics**

EffectBlend.EndShapePoint

Property **EndShapePoint** AS **ShapePoint**

EffectBlend

Description

The **EndShapePoint** property returns or sets the ending mapped node in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

You can specify which nodes CorelDRAW treats as the first node on start and end object's first nodes. These nodes determine how the intermediate objects are transformed from the start object to become the end object.

{button ,AL(`CLS_EffectBlend;FNC_EndShapePoint')} **Related Topics**

EffectBlend.Split

Function **Split**(ByVal **StepNo** AS Long) AS **Shape**

EffectBlend

Description

The **Split** method splits a blend at a specified step in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

You can split a blend to create a compound blend. The object you choose to split the blend becomes the end object for one component in the blend and the start object for the other. Thus, the original blend is split into two components.

Parameters	Description
StepNo	Specifies the step in the blend where the split occurs in the effect.

{button ,AL(`CLS_EffectBlend;FNC_Split')} **Related Topics**

EffectBlend.FuseStart

Function **FuseStart**() AS Boolean

EffectBlend

Description

The **FuseStart** method fuses the start components of a split or a compound blend to create a single blend in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

If the selected blend and at least two of the compound blend's components share the same start or end object, a curved pointer appears. Use this pointer to click the intermediate object in the component blend that you want to fuse.

{button ,AL(`CLS_EffectBlend;FNC_FuseStart')} **Related Topics**

EffectBlend.FuseEnd

Function **FuseEnd**() AS Boolean

EffectBlend

Description

The **FuseEnd** method fuses the end components of a split or a compound blend to create a single blend in a blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

If the selected blend and at least two of the compound blend's components share the same start or end object, a curved pointer appears. Use this pointer to click the intermediate object in the component blend that you want to fuse.

{button ,AL(`CLS_EffectBlend;FNC_FuseEnd')} **Related Topics**

EffectBlend.CopyFrom

Function **CopyFrom**(ByRef **Source** AS **EffectBlend**) AS Boolean

EffectBlend

Description

The **CopyFrom** method copies the properties from one blend effect into another blend effect in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can create a blend in which the intermediate objects progress along an existing path and stretch the objects along the full length of the path. You can also blend along a path you draw freehand.

Parameters	Description
Source	The **Source** parameter specifies a blend effect and copies its effect properties into the active blend effect object in CorelDRAW.

{button ,AL(`CLS_EffectBlend;FNC_CopyFrom')} **Related Topics**

EffectControlPath properties

EffectControlPath Legend

- Application
- Effects
- Parent

EffectControlPath

Class **EffectControlPath**

Properties Referenced by

The **EffectControlPath** class defines the characteristics of an effect's control path in CorelDRAW. A control path is a curve that determines the shape of an Artistic Media stroke. To change the shape of the curve, you must edit the control path. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

A powerful feature of the Artistic Media tool is the ability to apply Artistic Media strokes to existing objects. The outline of the object will be used as the control path for the Artistic Media stroke. To make it easier to apply Artistic Media strokes, you can open the Artistic Media Docker and either drag and drop Artistic Media strokes or choose strokes and apply them to selected objects.

When you apply an Artistic Media stroke, the stroke appears in the Last Used list box. This makes it easier to find frequently used Artistic Media strokes.

When you edit an Artistic Media stroke, you edit the control path of the stroke. The stroke itself is a closed path following the control path. When you shape the control path, the stroke shifts according to how you edit the control path. If you want to edit the closed path directly, you must first separate it from the control path.

{button ,AL(`CLS_EffectControlPath')} **Related Topics**

EffectControlPath.Application

Property **Application** AS **Application**

EffectControlPath

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectControlPath;FNC_Application')} **Related Topics**

EffectControlPath.Parent

Property **Parent** AS **Effect**

EffectControlPath

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectControlPath;FNC_Parent')} **Related Topics**

EffectControlPath.Effects

Property **Effects** AS **Effects**

EffectControlPath

The **Effects** property returns a collection of Blend and TextOnPath effects in CorelDRAW. A blend is a special effect that is created by blending one object with another through a progression of shapes and colors. The following examples show a basic blend and a blend on a path. In both cases, the blended objects are the start and finish (i.e., bottom and top) objects in the progression.

You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square). CorelDRAW treats text fitted to a path as one object. You can also separate the text from the object. When you separate a text from a curved or closed path, the text retains the shape of the object to which it was fitted. Straightening reverts text to its original appearance.

The **Effects** property returns a Read-Only value.

{button ,AL(`CLS_EffectControlPath;FNC_Effects')} **Related Topics**

EffectExtrude properties

EffectExtrude Legend

- AngleX
- AngleY
- AngleZ
- Application
- BaseColor
- BevelAngle
- BevelColor
- BevelDepth

- BevelGroup
  - Depth

- ExtrudeGroup

- FaceShape

- FaceVisible
  - LightIntensity
  - LightPosition
  - LightPresent

- Parent
  - Shading
  - ShadingColor
  - ShowBevelOnly
  - Type
  - UseBevel
  - UseExtrudeColorForBevel
  - UseFullColorRange
  - VanishingPoint

EffectExtrude methods

EffectExtrude Legend

- CopyFrom
- Rotate
- SetBevel
- SetLight

EffectExtrude

Class **EffectExtrude**

Properties Methods Referenced by

The **EffectExtrude** class defines the characteristics of extrude effect objects and describes the look and behavior of the objects through its properties and methods.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

For vector extrusions, CorelDRAW projects points from the object and joins them to create extruded surfaces. These surfaces are projected toward a vanishing point, adding depth to the original object so that it appears three-dimensional. However, the vanishing points for Back Parallel and Front Parallel extrusions are infinite, therefore the extruded surfaces can never converge.

After you've created a vector extrusion, you can edit it, apply fills to all or various extruded surfaces, and add light sources to enhance the effect of the extrusion.

You can also apply bitmap extrusions to objects you create in CorelDRAW. Applying bitmap extrusions lets you work with objects in three dimensions. Once you've created a bitmap extrusion, you can apply bevels, Ambient and Point lights, and texture fills. By rendering bitmap extrusions, CorelDRAW creates a two-dimensional bitmap.

{button ,AL(`CLS_EffectExtrude')} **Related Topics**

EffectExtrude.Application

Property **Application** AS **Application**

EffectExtrude

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_Application')} **Related Topics**

EffectExtrude.Parent

Property **Parent** AS **Effect**

EffectExtrude

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_Parent')} **Related Topics**

cdrExtrudeSmallBack=0

cdrExtrudeSmallFront=1

cdrExtrudeBigBack=2

cdrExtrudeBigFront=3

cdrExtrudeBackParallel=4

cdrExtrudeFrontParallel=5

EffectExtrude.Type

Property **Type** AS **cdrExtrudeType**

EffectExtrude

Description

The **Type** property returns or sets the extrude type in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns cdrExtrudeType.

{button ,AL(`CLS_EffectExtrude;FNC_Type')} **Related Topics**

EffectExtrude.VanishingPoint

Property **VanishingPoint** AS **ExtrudeVanishingPoint**

EffectExtrude

Description

The **VanishingPoint** property returns or set the vanishing point in an extrude effect in CorelDRAW. A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

{button ,AL(`CLS_EffectExtrude;FNC_VanishingPoint')} **Related Topics**

EffectExtrude.Depth

Property **Depth** AS Long

EffectExtrude

Description

The **Depth** property returns or sets the extrude depth in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_Depth')} **Related Topics**

EffectExtrude.AngleX

Property **AngleX** AS Double

EffectExtrude

Description

The **AngleX** property returns or sets the X axis extrude angle in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property is a percentage that rotates the extrusion along the x-axis.

{button ,AL(`CLS_EffectExtrude;FNC_AngleX')} **Related Topics**

EffectExtrude.AngleY

Property **AngleY** AS Double

EffectExtrude

Description

The **AngleY** property returns or sets the Y axis extrude angle in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property is a percentage that rotates the extrusion along the y-axis.

{button ,AL(`CLS_EffectExtrude;FNC_AngleY')} **Related Topics**

EffectExtrude.AngleZ

Property **AngleZ** AS Double

EffectExtrude

Description

The **AngleZ** property returns or sets the Z axis extrude angle in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property is a percentage that rotates the extrusion along the z-axis.

{button ,AL(`CLS_EffectExtrude;FNC_AngleZ')} **Related Topics**

cdrExtrudeObjectFill=0

cdrExtrudeSolidFill=1

cdrExtrudeColorShading=2

EffectExtrude.Shading

Property **Shading** AS **cdrExtrudeShading**

EffectExtrude

Description

The **Shading** property returns or sets the shading type in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns cdrExtrudeShading.

{button ,AL(`CLS_EffectExtrude;FNC_Shading')} **Related Topics**

EffectExtrude.BaseColor

Property **BaseColor** AS **Color**

EffectExtrude

Description

The **BaseColor** property returns or sets the solid fill color or starting shading color in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

{button ,AL(`CLS_EffectExtrude;FNC_BaseColor')} **Related Topics**

EffectExtrude.ShadingColor

Property **ShadingColor** AS **Color**

EffectExtrude

Description

The **ShadingColor** property returns or sets the ending shading color in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

{button ,AL(`CLS_EffectExtrude;FNC_ShadingColor')} **Related Topics**

EffectExtrude.UseBevel

Property **UseBevel** AS Boolean

EffectExtrude

Description

The **UseBevel** property returns or sets a True or False value that indicates whether or not to use a bevel in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_UseBevel')} **Related Topics**

EffectExtrude.ShowBevelOnly

Property **ShowBevelOnly** AS Boolean

EffectExtrude

Description

The **ShowBevelOnly** property returns or sets a True or False value that indicates whether or not to show only the bevel in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_ShowBevelOnly')} **Related Topics**

EffectExtrude.BevelDepth

Property **BevelDepth** AS Double

EffectExtrude

Description

The **BevelDepth** property returns or sets the bevel depth in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_BevelDepth')} **Related Topics**

EffectExtrude.BevelAngle

Property **BevelAngle** AS Double

EffectExtrude

Description

The **BevelAngle** returns or sets the bevel angle in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_BevelAngle')} **Related Topics**

EffectExtrude.UseExtrudeColorForBevel

Property **UseExtrudeColorForBevel** AS Boolean

EffectExtrude

Description

The **UseExtrudeColorForBevel** property returns or sets a True or False value that indicates whether or not to use the extrude color for the bevel in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_UseExtrudeColorForBevel')} **Related Topics**

EffectExtrude.BevelColor

Property **BevelColor** AS **Color**

EffectExtrude

Description

The **BevelColor** property returns or sets the bevel color in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

{button ,AL(`CLS_EffectExtrude;FNC_BevelColor')} **Related Topics**

EffectExtrude.LightPresent

Property **LightPresent**(ByVal **Index** AS Long) AS Boolean

EffectExtrude

Description

The **LightPresent** property returns or sets a True or False value that indicates whether or not there is light present at a given index in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
Index	The **Index** parameter specifies the light source used in an extrude effect.You can enhance an extrusion's effect and fill attributes by applying as many as three light sources. You can also remove applied light sources.

{button ,AL(`CLS_EffectExtrude;FNC_LightPresent')} **Related Topics**

cdrLightFrontTopLeft=0

cdrLightFrontTop=1

cdrLightFrontTopRight=2

cdrLightFrontLeft=3

cdrLightFrontCenter=4

cdrLightFrontRight=5

cdrLightFrontBottomLeft=6

cdrLightFrontBottom=7

cdrLightFrontBottomRight=8

cdrLightBackTopLeft=9

cdrLightBackTop=10

cdrLightBackTopRight=11

cdrLightBackRight=14

cdrLightBackBottomRight=17

EffectExtrude.LightPosition

Property **LightPosition**(ByVal **Index** AS Long) AS **cdrExtrudeLightPosition**

EffectExtrude

Description

The **LightPosition** property returns or sets the light position of an extrude effect at a given index in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns cdrExtrudeLightPosition.

Parameters	Description
Index	The **Index** parameter specifies the light source used in an extrude effect.You can enhance an extrusion's effect and fill attributes by applying as many as three light sources. You can also remove applied light sources.

{button ,AL(`CLS_EffectExtrude;FNC_LightPosition')} **Related Topics**

EffectExtrude.LightIntensity

Property **LightIntensity**(ByVal **Index** AS Long) AS Long

EffectExtrude

Description

The **LightIntensity** property returns or sets the light intensity of an extrude effect at a given index in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
Index	The **Index** parameter specifies the light source used in an extrude effect. You can enhance an extrusion's effect and fill attributes by applying as many as three light sources. You can also remove applied light sources.

{button ,AL(`CLS_EffectExtrude;FNC_LightIntensity')} **Related Topics**

EffectExtrude.UseFullColorRange

Property **UseFullColorRange** AS Boolean

EffectExtrude

Description

The **UseFullColorRange** property returns or sets a True or False value that indicates whether or not to use the full color range in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

{button ,AL(`CLS_EffectExtrude;FNC_UseFullColorRange')} **Related Topics**

EffectExtrude.FaceVisible

Property **FaceVisible** AS Boolean

EffectExtrude

Description

The **FaceVisible** property returns a True or False value that indicates whether or not the control object face is visible in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_FaceVisible')} **Related Topics**

EffectExtrude.FaceShape

Property **FaceShape** AS **Shape**

EffectExtrude

Description

The **FaceShape** property returns the shape of the control object face in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_FaceShape')} **Related Topics**

EffectExtrude.BevelGroup

Property **BevelGroup** AS **Shape**

EffectExtrude

Description

The **BevelGroup** property returns the bevel group in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_BevelGroup')} **Related Topics**

EffectExtrude.ExtrudeGroup

Property **ExtrudeGroup** AS **Shape**

EffectExtrude

Description

The **ExtrudeGroup** property returns the extrude group in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectExtrude;FNC_ExtrudeGroup')} **Related Topics**

EffectExtrude.Rotate

Sub **Rotate**(ByVal **AngleX** AS Double, ByVal **AngleY** AS Double, ByVal **AngleZ** AS Double)

EffectExtrude

Description

The **Rotate** method rotates an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
AngleX	Sets the degree of the rotation along the x axis.
AngleY	Sets the degree of the rotation along the y axis.
AngleZ	Sets the degree of the rotation along the z axis.

{button ,AL(`CLS_EffectExtrude;FNC_Rotate')} **Related Topics**

EffectExtrude.SetBevel

Sub **SetBevel**(ByVal **Depth** AS Double, ByVal **Angle** AS Double, ByVal **ShowBevelOnly** AS Boolean)

EffectExtrude

Description

The **SetBevel** method sets the properties for the bevel in an extrude effect in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Beveling creates the illusion that an object's edges are cut at an angle other than 90 degrees. You specify the appropriate angle and depth values for the size of the object being cut. The illusion itself is created through the addition of objects on top of the control object (the object you extruded). These objects work together to give the object a three-dimensional look.

Parameters	Description
Depth	Sets the depth percentage of the bevel in an extrude effect.
Angle	Sets the degree of the bevel in the extrude effect.
ShowBevelOnly	Sets a True or False value that indicates whether or not to show only the bevel in the extrude effect.

{button ,AL(`CLS_EffectExtrude;FNC_SetBevel')} **Related Topics**

EffectExtrude.SetLight

Sub **SetLight**(ByVal **Index** AS Long, ByVal **Position** AS **cdrExtrudeLightPosition**, ByVal **LightIntensity** AS Long)

EffectExtrude

Description

The **SetLight** method sets the light of an extrude effect at a given index in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
Index	Specifies the index in the extrude effect where the light property is set.
Position	Specifies the position of the light property in the extrude effect. This value returns cdrExtrudeLightPosition.
LightIntensity	Sets the light intensity in the extrude effect.

{button ,AL(`CLS_EffectExtrude;FNC_SetLight')} **Related Topics**

EffectExtrude.CopyFrom

Sub **CopyFrom**(ByRef **Source** AS **EffectExtrude**)

EffectExtrude

Description

The **CopyFrom** method copies the extrude effect properties from one extrude effect and applies them to the extrude effect of the active shape in CorelDRAW. Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Applying extrusions gives a two-dimensional object the illusion of depth. There are two types of extrusions - vector and bitmap.

Parameters	Description
Source	The **Source** parameter identifies the extrusion effect who's properties are copied into another extrusion effect in CorelDRAW.

{button ,AL(`CLS_EffectExtrude;FNC_CopyFrom')} **Related Topics**

ExtrudeVanishingPoint properties

ExtrudeVanishingPoint Legend

- Application
- Effects
- Parent
  - PositionX
  - PositionY
  - Type

ExtrudeVanishingPoint methods

ExtrudeVanishingPoint Legend

Share

ExtrudeVanishingPoint

Class **ExtrudeVanishingPoint**

Properties Methods Referenced by

The **ExtrudeVanishingPoint** class defines the characteristics of vanishing points in extrude effect objects and describes the look and behavior of the objects through its properties and methods.

A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

{button ,AL(`CLS_ExtrudeVanishingPoint')} **Related Topics**

ExtrudeVanishingPoint.Application

Property **Application** AS **Application**

ExtrudeVanishingPoint

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_Application')} **Related Topics**

ExtrudeVanishingPoint.Parent

Property **Parent** AS **EffectExtrude**

ExtrudeVanishingPoint

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_Parent')} **Related Topics**

cdrVPLockedToShape=0

cdrVPLockedToPage=1

cdrVPShared=2

ExtrudeVanishingPoint.Type

Property **Type** AS **cdrExtrudeVPType**

ExtrudeVanishingPoint

Description

The **Type** property returns or set the vanishing point type in an extrude effect in CorelDRAW.A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

This property returns a value of ExtrudeVPType.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_Type')} **Related Topics**

ExtrudeVanishingPoint.PositionX

Property **PositionX** AS Double

ExtrudeVanishingPoint

Description

The **PositionX** property returns or set the horizontal coordinate of a vanishing point in an extrude effect object in CorelDRAW. A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_PositionX')} **Related Topics**

ExtrudeVanishingPoint.PositionY

Property **PositionY** AS Double

ExtrudeVanishingPoint

Description

The **PositionY** property returns or set the vertical coordinateof a vanishing point in an extrude effect object in CorelDRAW. A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_PositionY')} **Related Topics**

ExtrudeVanishingPoint.Effects

Property **Effects** AS **Effects**

ExtrudeVanishingPoint

Description

The **Effects** property returns a value associated with a collection of extrude objects that share the same vanishing point in CorelDRAW. A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

This value returns a Read-Only value.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_Effects')} **Related Topics**

ExtrudeVanishingPoint.Share

Function **Share**(ByRef **Source** AS **EffectExtrude**) AS Boolean

ExtrudeVanishingPoint

Description

The **Share** property returns or sets a True or False value that indicates whether or not a vanishing point is shared in CorelDRAW. You can copy a vanishing point to another object or have multiple vector extrusions share the same vanishing point.

A vanishing point is a marker that appears when you select an extrusion or an object to which perspective has been added. With an extrusion, the vanishing point marker indicates the depth (parallel extrusion) or the point at which the extruded surfaces would meet if extended (perspective extrusion). With the Perspective effect, the marker indicates the point (or points) at which the nonparallel lines would meet.

Extrusion is a feature that lets you give objects a three-dimensional (3D) look by creating the illusion of depth. You can change the direction and depth of the extrude, the position of the vanishing point, its placement in 3D space, and its color.

Parameters	Description
Source	The **Source** parameter identifies the extrusion that has the vanishing point tha tyou want shared.

{button ,AL(`CLS_ExtrudeVanishingPoint;FNC_Share')} **Related Topics**

EffectEnvelope properties

EffectEnvelope Legend

- Application
  - Container
  - KeepLines
  - Mode
- Parent

EffectEnvelope methods

EffectEnvelope Legend

- CopyFrom
- CreateFrom
- Select

EffectEnvelope

Class **EffectEnvelope**

Properties Methods Referenced by

The **EffectEnvelope** class defines the characteristics of an envelope effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

After you've applied an envelope to an object, you can edit the envelope in a variety of ways you can change its editing mode, apply a new mapping mode, convert envelope segments from lines to curves and vice versa, and edit an envelope's node. You can add, remove, move, or change the shape of envelope nodes.

The editing modes control the shape of the envelope itself. Three of these modes - Straight Line, Single Arc, and Double Arc - let you drag a node or control point horizontally or vertically to change the shape of one side of the envelope. The fourth mode, Unconstrained, lets you drag a node in any direction to make more dramatic changes. In addition, this mode has control points for each node, which let you make precise adjustments to get the exact envelope shape you want.

By applying a new mapping mode, you change how CorelDRAW fits the object to the envelope, not the shape of the envelope itself. There are four mapping modes: Horizontal, Original, Putty, and Vertical. A fifth mode, Text, appears if you're using the envelope to reshape Paragraph text.

As with object segments, envelope segments can be converted from curves to straight lines or from straight lines to curves. By changing the segment's type, you change the way the envelope reacts when it's edited.

When you add nodes to an envelope, you can make minute adjustments to give the envelope a more complex shape. You can simplify the envelope's shape by removing nodes. The most basic method of editing an envelope is to move its nodes. For example, you can move adjacent nodes an equal distance in the same direction.

You can change a node's type, which changes the way the envelope segments on either side pass through the node. Changing the node type changes the shape of the envelope, which in turn changes the effect the envelope has on the object.

By removing an envelope from an object, you return the object to its shape prior to applying the envelope.

{button ,AL(`CLS_EffectEnvelope')} **Related Topics**

EffectEnvelope.Application

Property **Application** AS **Application**

EffectEnvelope

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
With ActiveShape.Effect.Envelope
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_Application')} **Related Topics**

EffectEnvelope.Parent

Property **Parent** AS **Effect**

EffectEnvelope

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example counts the number of effects in the envelope effect's parent **Effects** collection:

```
Sub EffectCount()
With ActiveShape.Effect.Envelope
    MsgBox .Parent.Parent.Count
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_Parent')} **Related Topics**

EffectEnvelope.Container

Property **Container** AS **Curve**

EffectEnvelope

Description

The **Container** property returns or sets the container for the envelope effect in CorelDRAW. The container is the curve object that contains the envelope effect. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

Example

The following code example counts the number of nodes on the shape object that contains the envelope effect:

```
Sub EffectNode()
With ActiveShape.Effect.Envelope
    MsgBox .Container.Nodes.Count
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_Container')} **Related Topics**

cdrEnvelopeHorizontal=0

cdrEnvelopeOriginal=1

cdrEnvelopePutty=2

cdrEnvelopeVertical=3

EffectEnvelope.Mode

Property **Mode** AS **cdrEnvelopeMode**

EffectEnvelope

Description

The **Mode** property returns or sets the mapping mode type for the envelope effect in CorelDRAW. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

By applying a new mapping mode, you change how CorelDRAW fits the object to the envelope, not the shape of the envelope itself. There are four mapping modes: Horizontal, Original, Putty, and Vertical. A fifth mode, Text, appears if you're using the envelope to reshape Paragraph text.

The **Mode** property returns a value of cdrEnvelopeMode.

Example

The following code example sets the mapping mode of the active shape's envelope effect to a horizontal mapping mode, which fits the shape horizontally to the envelope effect:

```
Sub EffectMode()
With ActiveShape.Effect.Envelope
    .Mode = cdrEnvelopeHorizontal
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_Mode')} **Related Topics**

EffectEnvelope.KeepLines

Property **KeepLines** AS Boolean

EffectEnvelope

Description

The **KeepLines** property returns or sets a True or False value that indicates if an envelope effect keeps its lines straight or converts them to curves in CorelDRAW. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

If the **KeepLines** property is True, the envelope effect keeps its lines straight - it does not convert the lines to curves.

Example

The following code example sets the **KeepLines** property to True, which keeps the lines in the envelope effect object as straight lines, not converted curve lines:

```
Sub Lines()
With ActiveShape.Effect.Envelope
    .KeepLines = True
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_KeepLines')} **Related Topics**

EffectEnvelope.Select

Sub **Select**(ByVal **PresetIndex** AS Long)

EffectEnvelope

Description

The **Select** method selects a preset envelope effect in CorelDRAW and applies that effect to a shape object. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

In order to use the **Select** method, you must pass the **PresetIndex** value as a parameter:

Parameters	Description
PresetIndex	The **PresetIndex** parameter is an index number that uniquely identifies a preset envelope effect in CorelDRAW.

Example

The following code example selects the second preset envelope effect in CorelDRAw and applies it to the active shape:

```
Sub Preset()
With ActiveShape.Effect.Envelope
    .Select (2)
End With
End Sub
```

{button ,AL(`CLS_EffectEnvelope;FNC_Select')} **Related Topics**

EffectEnvelope.CopyFrom

Sub **CopyFrom**(ByRef **Source** AS **EffectEnvelope**)

EffectEnvelope

Description

The **CopyFrom** method copies the properties from one envelope effect into another envelope effect in CorelDRAW. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

If you've applied an effect to the object since you applied the envelope, you won't be able to copy the envelope. You can apply the same envelope effect to several different objects.

In order to use the **CopyFrom** method, you must pass the **Source** value as a parameter:

Parameters	Description
Source	The **Source** parameter specifies an envelope effect and copies its effect properties into the active envelope effect object in CorelDRAW.

{button ,AL(`CLS_EffectEnvelope;FNC_CopyFrom')} **Related Topics**

EffectEnvelope.CreateFrom

Function **CreateFrom**(ByRef **Shape** AS **Shape**) AS Boolean

EffectEnvelope

Description

The **CreateFrom** method sets a True or False value that creates a new envelope effect from a specified shape object in CorelDRAW. An envelope effect is a feature that allows you to distort the shape of an object. Distortion is created by dragging nodes on an imaginary box (the envelope), which in turn distorts the object. You can apply a basic or a preset envelope. You can also copy an envelope from an object in your drawing and apply it to another object.

In order to use the **CreateFrom** method, you must pass the **Shape** value as a parameter:

Parameters	Description
Shape	The **Shape** parameter specifies the shape object to create a new envelope effect in CorelDRAW. A shape is any object that can be displayed as several variations of a rectangle or a circle.

{button ,AL(`CLS_EffectEnvelope;FNC_CreateFrom')} **Related Topics**

EffectTextOnPath properties

EffectTextOnPath Legend

- Application
  - DistanceFromPath
  - Offset
  - Orientation

- Parent
  - Path
  - Placement
  - PlaceOnOtherSide
  - Quadrant
  - Text
  - VertPlacement

EffectTextOnPath

Class **EffectTextOnPath**

Properties Referenced by

The **EffectTextOnPath** class defines the characteristics of text-on-path effect objects and describes the look and behavior of the objects through its properties and methods.

You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square). Once you've fit text to a path, you can specify the orientation of characters relative to the path. This allows you to create the impression that letters are standing upright. You can also rotate individual characters to follow the contours of the path.

You can also adjust the spacing between the text and path, specify the vertical position and vertical orientation of the text using the characters' baseline, ascender, descender, or center point, specify the horizontal position of text along the path and position text to the opposite side of the path

CorelDRAW treats text fitted to a path as one object. You can also separate the text from the object. When you separate a text from a curved or closed path, the text retains the shape of the object to which it was fitted. Straightening reverts text to its original appearance.

{button ,AL(`CLS_EffectTextOnPath')} **Related Topics**

EffectTextOnPath.Application

Property **Application** AS **Application**

EffectTextOnPath

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectTextOnPath;FNC_Application')} **Related Topics**

EffectTextOnPath.Parent

Property **Parent** AS **Effect**

EffectTextOnPath

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectTextOnPath;FNC_Parent')} **Related Topics**

EffectTextOnPath.Text

Property **Text** AS **Shape**

EffectTextOnPath

Description

The **Text** property returns or sets the text object used in a text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

You can't fit text to the path of another text object.

{button ,AL(`CLS_EffectTextOnPath;FNC_Text')} **Related Topics**

EffectTextOnPath.Path

Property **Path** AS **Shape**

EffectTextOnPath

Description

The **Path** property returns or sets the path object used in a text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

The path is a shape object - the text will follow the path of the shape to create the text on path effect in CorelDRAW.

You can't fit text to the path of another text object.

{button ,AL(`CLS_EffectTextOnPath;FNC_Path')} **Related Topics**

EffectTextOnPath.DistanceFromPath

Property **DistanceFromPath** AS Double

EffectTextOnPath

Description

The **DistanceFromPath** property returns or sets the vertical distance between the text and its path object in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

You can adjust the spacing between the text and path.

{button ,AL(`CLS_EffectTextOnPath;FNC_DistanceFromPath')} **Related Topics**

EffectTextOnPath.Offset

Property **Offset** AS Double

EffectTextOnPath

Description

The **Offset** property returns or sets the offset distance of the text object from its path object in a text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

Offset refers to a movement out of line with another object. An offset object is distanced from another specified object and this distance is the offset value.

{button ,AL(`CLS_EffectTextOnPath;FNC_Offset')} **Related Topics**

EffectTextOnPath.Orientation

Property **Orientation** AS **cdrFittedOrientation**

EffectTextOnPath

Description

The **Orientation** property returns or sets the text orientation type of a text on path effect object in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

Orientation is the direction in which a document is displayed on the page. A page oriented so that the horizontal dimension is greater than the vertical dimension has a landscape orientation. A page whose vertical dimension is greater than the horizontal dimension has a portrait orientation.

This value returns cdrFittedOrientation.

{button ,AL(`CLS_EffectTextOnPath;FNC_Orientation')} **Related Topics**

cdrRotateOrientation=0

cdrVerticalSkewOrientation=1

cdrHorizontalSkewOrientation=2

cdrUprightOrientation=3

cdrLeftPlacement=0

cdrRightPlacement=1

cdrCenterPlacement=2

EffectTextOnPath.Placement

Property **Placement** AS **cdrFittedPlacement**

EffectTextOnPath

Description

The **Placement** property returns or set the text placement in a text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

The property returns a value of cdrFittedPlacement.

{button ,AL(`CLS_EffectTextOnPath;FNC_Placement')} **Related Topics**

EffectTextOnPath.PlaceOnOtherSide

Property **PlaceOnOtherSide** AS Boolean

EffectTextOnPath

Description

The **PlaceOnOtherSide** property returns a True or False value that indicates whether or not text is placed on the opposite side of the path object in a text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

A value of True places the text on the opposite side of the path object.

{button ,AL(`CLS_EffectTextOnPath;FNC_PlaceOnOtherSide')} **Related Topics**

EffectTextOnPath.Quadrant

Property **Quadrant** AS **cdrFittedQuadrant**

EffectTextOnPath

Description

The **Quadrant** property returns or sets the text quadrant of the text on path effect in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

This property returns cdrFittedQuadrant.

{button ,AL(`CLS_EffectTextOnPath;FNC_Quadrant')} **Related Topics**

cdrLeftQuadrant=0

cdrRightQuadrant=1

cdrTopQuadrant=2

cdrBottomQuadrant=3

EffectTextOnPath.VertPlacement

Property **VertPlacement** AS **cdrFittedVertPlacement**

EffectTextOnPath

Description

The **VertPlacement** property returns or sets the vertical text placement of a text on path effect object in CorelDRAW. You can position Artistic text along the path of an open object (e.g., a line) or closed object (e.g., a square).

The vertical placement property aligns the baseline of the text with the path.

This property returns cdrFittedVertPlacement.

{button ,AL(`CLS_EffectTextOnPath;FNC_VertPlacement')} **Related Topics**

cdrCustomVertPlacement=0

cdrBaselineVertPlacement=1

cdrAscenderVertPlacement=2

cdrDescenderVertPlacement=3

cdrCenterVertPlacement=4

EffectDropShadow properties

EffectDropShadow Legend

- Application
  - Color
  - Fade
  - Feather
  - FeatherEdge
  - FeatherType
  - OffsetX
  - OffsetY
  - Opacity
- Parent
  - PerspectiveAngle
  - PerspectiveStretch
  - Type

EffectDropShadow methods

EffectDropShadow Legend

SetOffset

EffectDropShadow

Class **EffectDropShadow**

Properties Methods Referenced by

The **EffectDropShadow** class defines the characteristics of drop shadow effect objects and describes the look and behavior of the objects through its properties and methods.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

Additionally, you can edit drop shadows to customize their effects. For example, you can change the color of a drop shadow.

When you apply a drop shadow to an object, you create a linked group. The object to which you applied the drop shadow is the control object. Changes you make to the control object are reflected in the drop shadow. For example, if you change the size of the of the control object, the size drop shadow changes to reflect the control object's size. However, you can separate the drop shadow, which is a bitmap, and the control object to make them separate objects.

By copying or cloning, you can apply drop shadows to other objects. Copying a drop shadow transfers the drop shadow's attributes to another selected object. Cloning also transfers a drop shadow's attributes to a selected object; however, changes made to the original drop shadow (also called the master) are applied to the clone. Additionally, you can't edit the cloned drop shadow's settings; any changes must be made to the master drop shadow.

If you don't like a drop shadow effect that you've added, you can remove it from the object. You can also remove the object and keep the drop shadow.

{button ,AL(`CLS_EffectDropShadow')} **Related Topics**

EffectDropShadow.Application

Property **Application** AS **Application**

EffectDropShadow

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectDropShadow;FNC_Application')} **Related Topics**

EffectDropShadow.Parent

Property **Parent** AS **Effect**

EffectDropShadow

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectDropShadow;FNC_Parent')} **Related Topics**

EffectDropShadow.OffsetX

Property **OffsetX** AS Double

EffectDropShadow

Description

The **OffsetX** property returns the horizontal offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_OffsetX')} **Related Topics**

EffectDropShadow.OffsetY

Property **OffsetY** AS Double

EffectDropShadow

Description

The **OffsetY** property returns the vertical offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_OffsetY')} **Related Topics**

EffectDropShadow.Opacity

Property **Opacity** AS Long

EffectDropShadow

Description

The **Opacity** property returns the opacity value of a drop shadow effect in CorelDRAW. Opacity determines the intensity of a drop shadow effect. You can type values between 0 and 100. Low values create a less opaque drop shadow, while high values create a more opaque drop shadow.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_Opacity')} **Related Topics**

EffectDropShadow.Feather

Property **Feather** AS Long

EffectDropShadow

Description

The **Feather** property returns the feather length for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_Feather')} **Related Topics**

EffectDropShadow.FeatherType

Property **FeatherType** AS **cdrFeatherType**

EffectDropShadow

Description

The **FeatherType** property returns the feather type for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

This property returns cdrFeatherType.

{button ,AL(`CLS_EffectDropShadow;FNC_FeatherType')} **Related Topics**

cdrFeatherInside=0

cdrFeatherMiddle=1

cdrFeatherOutside=2

cdrFeatherAverage=3

cdrEdgeLinear=0

cdrEdgeSquared=1

cdrEdgeFlat=2

cdrEdgeInverseSquared=3

cdrEdgeMesa=4

cdrEdgeGaussian=5

EffectDropShadow.FeatherEdge

Property **FeatherEdge** AS **cdrEdgeType**

EffectDropShadow

Description

The **FeatherEdge** property returns the feather edge for a drop shadow effect in CorelDRAW. You can change a drop shadow's feathering edge style, the feathering direction, as well as the level of feathering. You can type values between 0 and 100. Low values create a more subtle feathering effect, while high values create a more pronounced effect.

By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

This property returns cdrEdgeType.

{button ,AL(`CLS_EffectDropShadow;FNC_FeatherEdge')} **Related Topics**

cdrDropShadowFlat=0

cdrDropShadowBottom=1

cdrDropShadowTop=2

cdrDropShadowLeft=3

cdrDropShadowRight=4

EffectDropShadow.Type

Property **Type** AS **cdrDropShadowType**

EffectDropShadow

Description

The Type property returns or sets the drop shadow type in a drop shadow effect in CorelDRAW.By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

This property returns cdrDropShadowType.

{button ,AL(`CLS_EffectDropShadow;FNC_Type')} **Related Topics**

EffectDropShadow.PerspectiveAngle

Property **PerspectiveAngle** AS Double

EffectDropShadow

Description

The **PersectiveAngle** returns or set the feather perspective angle in a drop shadow effect in CorelDRAW. By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_PerspectiveAngle')} **Related Topics**

EffectDropShadow.PerspectiveStretch

Property **PerspectiveStretch** AS Double

EffectDropShadow

Description

The **PerspectiveStretch** returns or set the feather perspective stretch value in a drop shadow effect in CorelDRAW. By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_PerspectiveStretch')} **Related Topics**

EffectDropShadow.Fade

Property **Fade** AS Long

EffectDropShadow

Description

The **Fade** property returns or set the fade value of a drop shadow effect in CorelDRAW. By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

You can't change the fade level of a drop shadow which has a Flat perspective.

{button ,AL(`CLS_EffectDropShadow;FNC_Fade')} **Related Topics**

EffectDropShadow.Color

Property **Color** AS **Color**

EffectDropShadow

Description

The **Color** property returns or sets the color of a drop shadow effect in CorelDRAW. By adding drop shadows, you create the illusion of depth between objects. You can add drop shadows to most objects (or groups of objects) you create using CorelDRAW, including Artistic text, Paragraph text, and bitmap images. However, you cannot add drop shadows to link groups such as blended objects, contoured objects, beveled objects, extruded objects, or other drop shadows.

{button ,AL(`CLS_EffectDropShadow;FNC_Color')} **Related Topics**

EffectDropShadow.SetOffset

Sub **SetOffset**(ByVal **OffsetX** AS Double, ByVal **OffsetY** AS Double)

EffectDropShadow

Description

The **Offset** property sets the offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object.

Parameters	Description
OffsetX	The **OffsetX** property returns the horizontal offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object.
OffsetY	The **OffsetY** property returns the vertical offset of a drop shadow effect in CorelDRAW. The offset value sets the distance between the drop shadow effect and its control object.

{button ,AL(`CLS_EffectDropShadow;FNC_SetOffset')} **Related Topics**

EffectLens properties

EffectLens Legend

- Application
  - Color
  - ColorMapPalette
  - FillColor
  - FromColor
- Frozen
  - Magnification
  - OutlineColor
  - PaletteRotation
- Parent
  - Rate
  - RemoveFace
- Shapes
  - ToColor
  - Type
  - UseFillColor
  - UseOutlineColor
  - UseViewPoint
  - ViewPointX
  - ViewPointY

EffectLens methods

EffectLens Legend

- Freeze
- Unfreeze
- Ungroup

EffectLens

Class **EffectLens**

Properties Methods Referenced by

The **EffectLens** class defines the characteristics of lens effect objects and describes the look and behavior of the objects through its properties and methods.

A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

When you apply a lens to an object, you change the way you perceive the objects behind it. You can choose amongst several types of lenses, each producing distinctive results. These results range from color alteration (as produced by heat map, inverting, and brightening lenses, for example) to distortion (as produced by magnifying and fish eye lenses). In each case, the lens changes the way you perceive the objects behind it, not the actual properties and attributes of those objects.

After you create a lens object, you can edit it to modify how you perceive the objects behind it.

By changing the viewpoint of a lens, you can display any portion of a drawing through the lens without moving it. The viewpoint represents the center point of what is being viewed through the lens. This point is indicated by an "X" in the Drawing Window. You can position the lens anywhere in the drawing, but it always shows the area around its viewpoint marker. For example, you can use the viewpoint marker on a Magnify lens to enlarge part of a map without obscuring any part of the map.

You can have CorelDRAW display a lens only where it overlaps other objects. As a result, the lens effect is not seen where the lens object covers blank space in the Drawing Window.

You can fix the contents of a lens by freezing it. You can then move the lens without changing what's displayed through the lens. Changes you make to the objects seen through the lens have no effect on the lens contents.

You can apply lenses to any closed-path object you create using CorelDRAW. You can also apply lenses to open-ended lines and curves, such as Paragraph text and Artistic text. Additionally, you can create lenses using objects imported from other applications, such as bitmaps. Once you create a lens you like, you can copy it to other objects in your drawing.

You can choose from the following lens types: Transparency, Magnify, Brighten, Invert, Color Limit, Color Add, Tinted Grayscale, Heat Map, Custom Color Map, Wireframe, and Fish Eye.

{button ,AL(`CLS_EffectLens')} **Related Topics**

EffectLens.Application

Property **Application** AS **Application**

EffectLens

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectLens;FNC_Application')} **Related Topics**

EffectLens.Parent

Property **Parent** AS **Effect**

EffectLens

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectLens;FNC_Parent')} **Related Topics**

EffectLens.Freeze

Sub **Freeze**()

EffectLens

Description

The **Freeze** method freezes a lens effect object in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

Frozen lenses redraw more quickly than non-frozen lenses. Once frozen, the contents of the object no longer interact with other objects on the screen.

{button ,AL(`CLS_EffectLens;FNC_Freeze')} **Related Topics**

EffectLens.Unfreeze

Sub **Unfreeze**()

EffectLens

Description

The **Unfreeze** method removes the freeze on a lens effect object in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

Frozen lenses redraw more quickly than non-frozen lenses. Once frozen, the contents of the object no longer interact with other objects on the screen.

{button ,AL(`CLS_EffectLens;FNC_Unfreeze')} **Related Topics**

EffectLens.Ungroup

Function **Ungroup**() AS **ShapeRange**

EffectLens

Description

The **Ungroup** method breaks up the selected group into its individual objects in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

Ungrouping is a command that causes a set of objects acting as one unit to behave as individual objects. If you have more than one sublevel of grouping, Ungroup breaks up one level of grouping at a time.

{button ,AL(`CLS_EffectLens;FNC_Ungroup')} **Related Topics**

EffectLens.Shapes

Property **Shapes** AS **Shapes**

EffectLens

Description

The **Shapes** property returns a value associated with the shapes of a frozen lens effect group in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

This property returns a Read-Only value.

{button ,AL(`CLS_EffectLens;FNC_Shapes')} **Related Topics**

EffectLens.Frozen

Property **Frozen** AS Boolean

EffectLens

Description

The **Frozen** property returns a True or False value that indicates if a lens is frozen in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

Frozen lenses redraw more quickly than non-frozen lenses. Once frozen, the contents of the object no longer interact with other objects on the screen.

The **Frozen** property returns a Read-Only value.

{button ,AL(`CLS_EffectLens;FNC_Frozen')} **Related Topics**

EffectLens.Type

Property **Type** AS **cdrLensType**

EffectLens

Description

The **Type** property returns or sets the lens type in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

This property returns cdrLensType.

{button ,AL(`CLS_EffectLens;FNC_Type')} **Related Topics**

cdrLensMagnify=0

cdrLensFishEye=1

cdrLensWireframe=2

cdrLensColorLimit=3

cdrLensColorAdd=4

cdrLensInvert=5

cdrLensBrighten=6

cdrLensTintedGrayscale=7

cdrLensHeatMap=8

cdrLensTransparency=9

cdrLensCustomColorMap=10

EffectLens.Rate

Property **Rate** AS Long

EffectLens

Description

The **Rate** property returns or sets the lens rate in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

The rate property applies to various lens effects, including Brighten, Color Add, Color Limit, and Fish eye lens effects. Rate refers to values between 0 and 100% that increase degree of a lens effect.

{button ,AL(`CLS_EffectLens;FNC_Rate')} **Related Topics**

EffectLens.Color

Property **Color** AS **Color**

EffectLens

Description

The **Color** property returns or sets the color that is used in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

Several lens effects use the color property, including the color add and color limit effects.

{button ,AL(`CLS_EffectLens;FNC_Color')} **Related Topics**

EffectLens.OutlineColor

Property **OutlineColor** AS **Color**

EffectLens

Description

The **OutlineColor** property returns or set the outline color in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

{button ,AL(`CLS_EffectLens;FNC_OutlineColor')} **Related Topics**

EffectLens.FillColor

Property **FillColor** AS **Color**

EffectLens

Description

The **FillColor** property returns or sets the lens fill color in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

{button ,AL(`CLS_EffectLens;FNC_FillColor')} **Related Topics**

EffectLens.FromColor

Property **FromColor** AS **Color**

EffectLens

Description

The **FromColor** property returns or sets the starting color in a Custom Color Map Lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

The Custom Color Map lens changes all the colors of objects behind it to a color range between any two colors you select. In addition to defining the range's start and end colors, you choose the progression between the two colors. The progression can follow a direct, forward, or reverse route through the color spectrum. Areas of the lens that do not cover other objects are filled with the color at the end of the color map.

{button ,AL(`CLS_EffectLens;FNC_FromColor')} **Related Topics**

EffectLens.ToColor

Property **ToColor** AS **Color**

EffectLens

Description

The **ToColor** property returns or sets the ending color in a Custom Color Map Lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

The Custom Color Map lens changes all the colors of objects behind it to a color range between any two colors you select. In addition to defining the range's start and end colors, you choose the progression between the two colors. The progression can follow a direct, forward, or reverse route through the color spectrum. Areas of the lens that do not cover other objects are filled with the color at the end of the color map.

{button ,AL(`CLS_EffectLens;FNC_ToColor')} **Related Topics**

EffectLens.UseOutlineColor

Property **UseOutlineColor** AS Boolean

EffectLens

Description

The **UseOutlineColor** property returns or sets a True or False value that indicates whether or not the outline color is used in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool flyout. When you create an object, CorelDRAW gives it a default outline that is a black solid fill.

{button ,AL(`CLS_EffectLens;FNC_UseOutlineColor')} **Related Topics**

EffectLens.UseFillColor

Property **UseFillColor** AS Boolean

EffectLens

Description

The **UseFillColor** property returns or sets a True or False value that indicates whether or not the fill color is used in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

{button ,AL(`CLS_EffectLens;FNC_UseFillColor')} **Related Topics**

EffectLens.ColorMapPalette

Property **ColorMapPalette** AS **cdrFountainFillBlendType**

EffectLens

Description

The **ColorMapPalette** property returns or sets the color map palette type in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

This value returns cdrFountainFillBlendType.

{button ,AL(`CLS_EffectLens;FNC_ColorMapPalette')} **Related Topics**

EffectLens.Magnification

Property **Magnification** AS Double

EffectLens

Description

The **Magnification** property returns or set the magnification level in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

{button ,AL(`CLS_EffectLens;FNC_Magnification')} **Related Topics**

EffectLens.UseViewPoint

Property **UseViewPoint** AS Boolean

EffectLens

Description

The **UseViewPoint** property returns or set s a True or False value that indicates whether or not a view point is used in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

The viewpoint represents the center point of what is being viewed through the lens. This point is marked by an "X" in the Drawing Window.

{button ,AL(`CLS_EffectLens;FNC_UseViewPoint')} **Related Topics**

EffectLens.ViewPointX

Property **ViewPointX** AS Double

EffectLens

Description

The **ViewPointX** property returns or sets the horizontal coordinate of a viewpoint in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

The viewpoint represents the center point of what is being viewed through the lens. This point is marked by an "X" in the Drawing Window.

{button ,AL(`CLS_EffectLens;FNC_ViewPointX')} **Related Topics**

EffectLens.ViewPointY

Property **ViewPointY** AS Double

EffectLens

Description

The **ViewPointY** property returns or sets the vertical coordinate of a viewpoint in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

A view in CorelDRAW allows you to view your drawing according to your needs. You may use several tools to change the way CorelDRAW displays objects in the application window. An active view refers to the view used in the current document.

The viewpoint represents the center point of what is being viewed through the lens. This point is marked by an "X" in the Drawing Window.

{button ,AL(`CLS_EffectLens;FNC_ViewPointY')} **Related Topics**

EffectLens.RemoveFace

Property **RemoveFace** AS Boolean

EffectLens

Description

The **RemoveFace** property returns or sets a True or False value that indicates whether or not to remove the lens face in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

The Remove Face option is available for color-altering lenses only and isn't available for Fish Eye and Magnify lenses.

{button ,AL(`CLS_EffectLens;FNC_RemoveFace')} **Related Topics**

EffectLens.PaletteRotation

Property **PaletteRotation** AS Long

EffectLens

Description

The **PaletteRotation** property returns or set the palette rotation amount in a lens effect in CorelDRAW. A lens lets you change the appearance of objects and, more significantly, the way you perceive objects behind. You can apply lens to almost any closed shape that has been created using the drawing tools in CorelDRAW.

{button ,AL(`CLS_EffectLens;FNC_PaletteRotation')} **Related Topics**

EffectPerspective properties

EffectPerspective Legend

- Application
  - HorizVanishingPointX
  - HorizVanishingPointY
- Parent
  - UseHorizVanishingPoint
  - UseVertVanishingPoint
  - VertVanishingPointX
  - VertVanishingPointY

EffectPerspective

Class **EffectPerspective**

Properties Referenced by

The **EffectPerspective** class defines the characteristics of perspective effect objects and describes the look and behavior of the objects through its properties and methods.

Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

After you add perspective to an object, you can further edit it to create the illusion you want.

Perspective is created by shortening one or two sides of an object. For one-point perspective, you shorten one side of an object so that it appears to recede in one direction. By shortening two sides, you get two-point perspective in which the object appears to recede in two directions.

When you've created a perspective you like, you can copy and apply it to one or more objects in your drawing. You can copy perspective to any object you've created using CorelDRAW, except Paragraph text.

Removing the perspective effect from objects restores the object to its original state. If you've applied an effect to the object since you applied perspective, you need to clear that effect before clearing perspective.

{button ,AL(`CLS_EffectPerspective')} **Related Topics**

EffectPerspective.Application

Property **Application** AS **Application**

EffectPerspective

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectPerspective;FNC_Application')} **Related Topics**

EffectPerspective.Parent

Property **Parent** AS **Effect**

EffectPerspective

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectPerspective;FNC_Parent')} **Related Topics**

EffectPerspective.UseHorizVanishingPoint

Property **UseHorizVanishingPoint** AS Boolean

EffectPerspective

Description

The **UseHorizVanishingPoint** property returns or sets a True or False value that indicates whether or not the horizontal vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_UseHorizVanishingPoint')} **Related Topics**

EffectPerspective.UseVertVanishingPoint

Property **UseVertVanishingPoint** AS Boolean

EffectPerspective

Description

The **UseVertVanishingPoint** property returns or sets a True or False value that indicates whether or not the vertical vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_UseVertVanishingPoint')} **Related Topics**

EffectPerspective.HorizVanishingPointX

Property **HorizVanishingPointX** AS Double

EffectPerspective

Description

The **HorizVanishingPointX** property returns or sets the X coordinate of the horizontal vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_HorizVanishingPointX')} **Related Topics**

EffectPerspective.HorizVanishingPointY

Property **HorizVanishingPointY** AS Double

EffectPerspective

Description

The **HorizVanishingPointY** property returns or sets the Y coordinate of the horizontal vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_HorizVanishingPointY')} **Related Topics**

EffectPerspective.VertVanishingPointX

Property **VertVanishingPointX** AS Double

EffectPerspective

Description

The **VertVanishingPointX** property returns or sets the X coordinate of the vertical vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_VertVanishingPointX')} **Related Topics**

EffectPerspective.VertVanishingPointY

Property **VertVanishingPointY** AS Double

EffectPerspective

Description

The **VertVanishingPointY** property returns or sets the Y coordinate of the vertical vanishing point is used in a perspective effect in CorelDRAW. Adding perspective to objects creates the illusion of distance and depth. Although objects are two-dimensional, applying one and two-point perspective adds another dimension to your drawing. By creating one-point perspective, you can make an object look like it's receding from view in one direction. By creating two-point perspective, on the other hand, you can make the object look like it's receding from view in two directions.

You can change the depth of a parallel extrusion and you can control the perspective of a perspective extrusion by moving the vanishing point. A vanishing point is defined by an X and Y coordinate.

{button ,AL(`CLS_EffectPerspective;FNC_VertVanishingPointY')} **Related Topics**

EffectContour properties

EffectContour Legend

- Application
  - ColorAcceleration
  - ColorBlendType
  - Direction
  - FillColor
  - FillColorTo
  - LinkAcceleration
  - Offset
  - OutlineColor
- Parent
  - SpacingAcceleration
  - Steps

EffectContour.ColorAcceleration

Property **ColorAcceleration** As Long

Description

The **ColorAcceleration** property returns or sets the rate of color acceleration in a contour effect in CorelDRAW. Higher numbers allow the colors to move quicker through the spectrum as they approach the end object.

You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_ColorAcceleration')} **Related Topics**

EffectContour.LinkAcceleration

Property **LinkAcceleration** As Boolean

Description

The **LinkAcceleration** property returns or sets a True or False value that links color acceleration to object spacing acceleration in a contour effect in CorelDRAW. You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_LinkAcceleration')} **Related Topics**

EffectContour.SpacingAcceleration

Property **SpacingAcceleration** As Long

Description

The **SpacingAcceleration** property returns or sets the rate of object acceleration in a contour effect in CorelDRAW. You can change the progression of the outline and fill colors, as well as the progression of the size of intermediate objects. You can link the rates of color and object acceleration so that they accelerate at the same rate.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_SpacingAcceleration')} **Related Topics**

EffectContour

Class **EffectContour**

Properties Referenced by

The **EffectContour** class defines the characteristics of contour effect objects and describes the look and behavior of the objects through its properties and methods.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

The spaces between contour lines are filled with colors that follow a progression from the original object to the last shape created. If there is a difference in color between the contour lines and the outline of the original object, a second progression occurs. You can modify both color progressions to get the look you want.

CorelDRAW provides three Contour effects that let you detect and accentuate the edges of images. You can set the level of edge detection, choose the type of edges, and define the color of the edges. The Contour effects are:

Edge Detect - detects the edges of items in an image and converts them to lines on a single-color background

Find Edges - locates edges in an image and lets you convert these edges to soft or solid lines

Trace Contour - traces image elements using a 16-color palette

When you apply contour lines to an object, you add a series of concentric lines or "steps" that radiate inside or outside of the object's borders. You create an effect like that created by contour lines on a topographical map. You can apply contours to any object you create using CorelDRAW, including shapes, lines, and curves. In addition, you'll find that you can create an array of interesting effects by applying contours to Artistic text.

Once you've created a contoured object you like, you can copy or clone its attributes to another object. When you copy a contour, the object to which you copied the contour takes on all contour-related settings of the original object, however the outline and fill attributes remain unaffected. The two objects have no connection and can be edited independently.

{button ,AL(`CLS_EffectContour')} **Related Topics**

EffectContour.Application

Property **Application** AS **Application**

EffectContour

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectContour;FNC_Application')} **Related Topics**

EffectContour.Parent

Property **Parent** AS **Effect**

EffectContour

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectContour;FNC_Parent')} **Related Topics**

cdrContourInside=0

cdrContourOutside=1

cdrContourToCenter=2

EffectContour.Direction

Property **Direction** AS **cdrContourDirection**

EffectContour

Description

The **Direction** property returns or sets the direction of a contour effect in CorelDRAW. Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

This property returns a value of cdrContourDirection.

{button ,AL(`CLS_EffectContour;FNC_Direction')} **Related Topics**

EffectContour.Offset

Property **Offset** AS Double

EffectContour

Description

The **Offset** property returns or sets the offset distance, in document units, between contour lines in a contour effect in CorelDRAW. This automatically adjusts the number of contour steps.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_Offset')} **Related Topics**

EffectContour.Steps

Property **Steps** AS Long

EffectContour

Description

The **Steps** property returns or sets the number of steps in a contour effect in CorelDRAW. The number of steps in a contour effect is linked to the number of lines that appear in the effect.

Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_Steps')} **Related Topics**

EffectContour.ColorBlendType

Property **ColorBlendType** AS **cdrFountainFillBlendType**

EffectContour

Description

The **ColorBlendType** property returns or set the color blend type of a fountain fill blend type in a contour effect in CorelDRAW. Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

This property returns cdrFountainFillBlendType.

{button ,AL(`CLS_EffectContour;FNC_ColorBlendType')} **Related Topics**

EffectContour.OutlineColor

Property **OutlineColor** AS **Color**

EffectContour

Description

The **OutlineColor** property returns or sets the contour outline color in CorelDRAW. Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

An outline is the line that defines the shape of an object.

{button ,AL(`CLS_EffectContour;FNC_OutlineColor')} **Related Topics**

EffectContour.FillColor

Property **FillColor** AS **Color**

EffectContour

Description

The **FillColor** property returns or sets the contour fill color in CorelDRAW. Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_FillColor')} **Related Topics**

EffectContour.FillColorTo

Property **FillColorTo** AS **Color**

EffectContour

Description

The **FillColorTo** property returns or sets the FillTo color of a contour effect in CorelDRAW. Contour refers to a special effect created by adding evenly spaced concentric lines inside or outside the borders of an object. These lines use the same shape as the outline of the original object, but they are smaller or larger depending on where they are created.

{button ,AL(`CLS_EffectContour;FNC_FillColorTo')} **Related Topics**

EffectDistortion properties

EffectDistortion Legend

- Application
  - OriginX
  - OriginY
- Parent
  - PushPull
  - Twister
  - Type
  - Zipper

EffectDistortion methods

EffectDistortion Legend

CenterDistortion

EffectDistortion

Class **EffectDistortion**

Properties Methods Referenced by

The **EffectDistortion** class defines the characteristics of distortion effect objects and describes the look and behavior of the objects through its properties and methods.

Distortion is a lack of proportionality in an image that results from defects in the optical system.

The interactive distortion tools let you apply interesting effects to objects. You can push or pull the nodes of objects you're distorting away from or towards their centers. By applying a zipper, you can create jagged, pointy outlines. You can also edit the effects of the Zipper distortion. You can also twist objects. By twisting an object, you wind the object around itself into a shape that resembles a coil.

Additionally, there are modifications that you can apply to all of the distortions that let you distort and customized them further. For example, you can change the center of a distortion.

When you're working with Push and Pull, Zipper, and Twister distortions, you can change the effect by moving the center of the distortion. You can also apply a different distortion to a distorted object. For example, you can apply a Twister distortion to an object to which you've applied a Zipper distortion.

You can apply each type of distortion effect to any object you create using CorelDRAW, including shapes, lines, curves, and Artistic text.

{button ,AL(`CLS_EffectDistortion')} **Related Topics**

EffectDistortion.Application

Property **Application** AS **Application**

EffectDistortion

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectDistortion;FNC_Application')} **Related Topics**

EffectDistortion.Parent

Property **Parent** AS **Effect**

EffectDistortion

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectDistortion;FNC_Parent')} **Related Topics**

EffectDistortion.Type

Property **Type** AS **cdrDistortionType**

EffectDistortion

Description

The **Type** property returns or sets the distortion type in a distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system. A distortion can be push pull, zipper or twister effects.

The **Type** property returns a value of cdrDistortionType.

{button ,AL(`CLS_EffectDistortion;FNC_Type')} **Related Topics**

cdrDistortionPushPull=0

cdrDistortionZipper=1

cdrDistortionTwister=2

EffectDistortion.OriginX

Property **OriginX** AS Double

EffectDistortion

Description

The **OriginX** property returns or sets the horizonal coordinateof a distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

This property is measured in document units.

{button ,AL(`CLS_EffectDistortion;FNC_OriginX')} **Related Topics**

EffectDistortion.OriginY

Property **OriginY** AS Double

EffectDistortion

Description

The **OriginY** property returns or sets the vertical coordinate of a distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

This property is measured in document units.

{button ,AL(`CLS_EffectDistortion;FNC_OriginY')} **Related Topics**

EffectDistortion.PushPull

Property **PushPull** AS **EffectPushPullDistortion**

EffectDistortion

Description

The **PushPull** property returns or sets a value associated with a push pull distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

When you apply a Push distortion to an object, the object's nodes are forced towards its center thereby rounding the object. The Pull distortion draws an object's nodes away from its center, making the object pointy.

When you create a Push or Pull distortion you like, you can copy it and apply it to other selected objects. If you don't like the distortion, you can remove it without deleting the object.

{button ,AL(`CLS_EffectDistortion;FNC_PushPull')} **Related Topics**

EffectDistortion.Zipper

Property **Zipper** AS **EffectZipperDistortion**

EffectDistortion

Description

The **Zipper** property returns or sets a value associated with a zipper distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

When you create a Zipper distortion you like, you can copy it and apply it to selected objects. If you don't like the distortion, you can remove it without deleting the object.

{button ,AL(`CLS_EffectDistortion;FNC_Zipper')} **Related Topics**

EffectDistortion.Twister

Property **Twister** AS **EffectTwisterDistortion**

EffectDistortion

Description

The **Twister** property returns or sets a value associated with a twister distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

The Twister distortion turns an object around itself. When you apply a Twister distortion to an object, the point at which you click becomes the center of the distortion, which can be also identified by a diamond-shaped handle. The center is fixed while one end of the object is wound around this fixed point. A dashed line, the Horizontal Line Of Origin, extends from the center of the object. This line measures the amount of Twister distortion in degrees you apply as you drag the rotation handle in either a clockwise or counterclockwise direction. One full rotation is 359 degrees.

{button ,AL(`CLS_EffectDistortion;FNC_Twister')} **Related Topics**

EffectDistortion.CenterDistortion

Sub **CenterDistortion**()

EffectDistortion

Description

The **CenterDistortion** method centers a distortion effect in the object that contains the distortion effect in CorelDRAW. Distortion is a lack of proportionality in an image that results from defects in the optical system.

{button ,AL(`CLS_EffectDistortion;FNC_CenterDistortion')} **Related Topics**

EffectPushPullDistortion properties

EffectPushPullDistortion Legend

Amplitude

Application

Parent

EffectPushPullDistortion

Class **EffectPushPullDistortion**

Properties Referenced by

The **EffectPushPullIDistortion** class defines the characteristics of push and pull effect objects and describes the look and behavior of the objects through its properties and methods.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

When you apply a Push distortion to an object, the object's nodes are forced towards its center thereby rounding the object. The Pull distortion draws an object's nodes away from its center, making the object pointy.

When you create a Push or Pull distortion you like, you can copy it and apply it to other selected objects. If you don't like the distortion, you can remove it without deleting the object.

{button ,AL(`CLS_EffectPushPullDistortion')} **Related Topics**

EffectPushPullDistortion.Application

Property **Application** AS **Application**

EffectPushPullDistortion

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectPushPullDistortion;FNC_Application')} **Related Topics**

EffectPushPullDistortion.Parent

Property **Parent** AS **EffectDistortion**

EffectPushPullDistortion

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectPushPullDistortion;FNC_Parent')} **Related Topics**

EffectPushPullDistortion.Amplitude

Property **Amplitude** AS Long

EffectPushPullDistortion

Description

The **Amplitude** property returns or sets a value that indicates the degree of distortion in a Push Pull Distortion effect. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects.

A distortion effect is a lack of proportionality in an image that results from defects in the optical system. The range of the distortion value is 1 to 200.

You can also create a Pull distortion by typing a value (from 1 to 200) in the Push And Pull Distortion Amplitude box on the Property Bar.

{button ,AL(`CLS_EffectPushPullDistortion;FNC_Amplitude')} **Related Topics**

EffectZipperDistortion properties

EffectZipperDistortion Legend

- Amplitude
- Application
- Frequency
- Local

- Parent
- Random
- Smooth

EffectZipperDistortion

Class **EffectZipperDistortion**

Properties Referenced by

The **EffectZipperDistortion** class defines the characteristics of a zipper distortion effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects.

When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

When you create a Zipper distortion you like, you can copy it and apply it to selected objects. If you don't like the distortion, you can remove it without deleting the object.

When you're working with Zipper distortions, you can customize the effect by rounding the zipper's points, adjusting the number of points per line segment, and emphasizing the distortion in a given area of the object.

When you're working with Push and Pull, Zipper, and Twister distortions, you can change the effect by moving the center of the distortion. You can also apply a different distortion to a distorted object. For example, you can apply a Twister distortion to an object to which you've applied a Zipper distortion.

You can also adjust the amplitude of the zipper effect by typing a value in the Zipper Distortion Amplitude box on the Property Bar. You can type values from 0 to 100. Higher values produce a more pronounced zipper distortion.

{button ,AL(`CLS_EffectZipperDistortion')} **Related Topics**

EffectZipperDistortion.Application

Property **Application** AS **Application**

EffectZipperDistortion

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
With ActiveShape.Effect.Distortion.Zipper
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_EffectZipperDistortion;FNC_Application')} **Related Topics**

EffectZipperDistortion.Parent

Property **Parent** AS **EffectDistortion**

EffectZipperDistortion

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Parent')} **Related Topics**

EffectZipperDistortion.Amplitude

Property **Amplitude** AS Long

EffectZipperDistortion

Description

The **Amplitude** property returns or sets the amplitude of a zipper distortion effect in CorelDRAW. When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

Amplitude determines the intensity of a zipper distortion effect. You can also adjust the amplitude of the zipper effect by typing a value in the Zipper Distortion Amplitude box on the Property Bar. You can type values from 0 to 100. Higher values produce a more pronounced zipper distortion.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Amplitude')} **Related Topics**

EffectZipperDistortion.Frequency

Property **Frequency** AS Long

EffectZipperDistortion

Description

The **Frequency** property returns or sets the frequency of a zipper distortion effect in CorelDRAW. When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

The **Frequency** property can have a range of values from 0 to 100. A higher frequency value creates more zipper points in the distortion effect.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Frequency')} **Related Topics**

EffectZipperDistortion.Random

Property **Random** AS Boolean

EffectZipperDistortion

Description

The **Random** property returns or sets a True or False value that indicates whether the zipper distortion effect is a random effect. When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

In a random zipper distortion, the horizontal and vertical points follow a haphazard course.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Random')} **Related Topics**

EffectZipperDistortion.Smooth

Property **Smooth** AS Boolean

EffectZipperDistortion

Description

The **Smooth** property returns or sets a True or False value that indicates whether the zipper distortion effect is a smooth effect. When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

A smooth distortion lets you smoothen the points of the Zipper distortion, instead of having them appear as jagged edges.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Smooth')} **Related Topics**

EffectZipperDistortion.Local

Property **Local** AS Boolean

EffectZipperDistortion

Description

The **Local** property returns or sets a True or False value that indicates whether the zipper distortion effect is a local effect. When you apply a Zipper distortion to an object, you create a series of points above and below the object's original outline. The outline follows these points to create a zipper effect. You can also create a random Zipper distortion, where the horizontal and vertical points follow a haphazard course.

A local distortion lets you emphasize the Zipper distortion in a specific area of the selected object.

{button ,AL(`CLS_EffectZipperDistortion;FNC_Local')} **Related Topics**

EffectTwisterDistortion properties

EffectTwisterDistortion Legend

Angle

Application

Parent

EffectTwisterDistortion

Class **EffectTwisterDistortion**

Properties Referenced by

The **EffectTwisterDistortion** class defines the characteristics of twister effect objects and describes the look and behavior of the objects through its properties and methods.

An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects. You can distort objects, add new elements to objects, and change the relationship between objects. Distortion is a lack of proportionality in an image that results from defects in the optical system.

The Twister distortion turns an object around itself. When you apply a Twister distortion to an object, the point at which you click becomes the center of the distortion, which can be also identified by a diamond-shaped handle. The center is fixed while one end of the object is wound around this fixed point. A dashed line, the Horizontal Line Of Origin, extends from the center of the object. This line measures the amount of Twister distortion in degrees you apply as you drag the rotation handle in either a clockwise or counterclockwise direction. One full rotation is 359 degrees.

CorelDRAW also lets you adjust the rotation of Twister distortions. For example, you can change the number of degrees you've rotated an object, as well as the direction of the rotation.

When you create a Twister distortion you like, you can copy it and apply it to other selected objects. If you don't like the distortion, you can remove it without deleting the object.

{button ,AL(`CLS_EffectTwisterDistortion')} **Related Topics**

EffectTwisterDistortion.Application

Property **Application** AS **Application**

EffectTwisterDistortion

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectTwisterDistortion;FNC_Application')} **Related Topics**

EffectTwisterDistortion.Parent

Property **Parent** AS **EffectDistortion**

EffectTwisterDistortion

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectTwisterDistortion;FNC_Parent')} **Related Topics**

EffectTwisterDistortion.Angle

Property **Angle** AS Double

EffectTwisterDistortion

Description

The **Angle** property returns or sets a value associated with the angle of the twister distortion effect in CorelDRAW. An effect is a process of altering or adjusting the appearance of an object in CorelDRAW. The special effects in CorelDRAW let you alter the appearance of objects.

A distortion effect is a lack of proportionality in an image that results from defects in the optical system. The range of the distortion value is 1 to 200.

The Twister distortion turns an object around itself. When you apply a Twister distortion to an object, the point at which you click becomes the center of the distortion, which can be also identified by a diamond-shaped handle. The center is fixed while one end of the object is wound around this fixed point. A dashed line, the Horizontal Line Of Origin, extends from the center of the object. This line measures the amount of Twister distortion in degrees you apply as you drag the rotation handle in either a clockwise or counterclockwise direction. One full rotation is 359 degrees.

{button ,AL(`CLS_EffectTwisterDistortion;FNC_Angle')} **Related Topics**

EffectArtistic properties

EffectArtistic Legend

- Application

- Parent

EffectArtistic

Class **EffectArtistic**

Properties Referenced by

The **EffectArtistic** class defines the characteristics of artistic effect objects and describes the look and behavior of the objects through its properties and methods.

The Artistic Media tool lets you apply various effects to a curve. For example, you can create curves that look like strokes from a calligraphic pen or a pressure-sensitive pen. As well, you can apply text, shapes, or images to a curve using the Artistic Media tool. If you need to change the position or shape of the curve, you can edit the Artistic Media stroke's control path.

The Artistic Media tool has five modes:

Preset mode - draws curves that change thickness based on preset line shapes you can choose from a list box on the Property Bar.

Brush mode - applies text or shapes to the curve when you draw it.

Object sprayer mode - applies a series of images to a curve when you draw it.

Calligraphic mode - draws curves that change thickness based on the direction of the curve. This creates an effect similar to that of a calligraphic pen.

Pressure-sensitive mode - draws curves that change thickness based on feedback from a pressure-sensitive pen or a keyboard.

{button ,AL(`CLS_EffectArtistic')} **Related Topics**

EffectArtistic.Application

Property **Application** AS **Application**

EffectArtistic

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

{button ,AL(`CLS_EffectArtistic;FNC_Application')} **Related Topics**

EffectArtistic.Parent

Property **Parent** AS **Pages**

EffectArtistic

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_EffectArtistic;FNC_Parent')} **Related Topics**

PatternCanvases properties

PatternCanvases Legend

- Count
- Item

PatternCanvases methods

PatternCanvases Legend

- Add
- Remove

PatternCanvases

Class **PatternCanvases**

Properties Methods Referenced by

The **PatternCanvases** class defines the characteristics of pattern canvas collection objects and describes the look and behavior of the collection objects through its properties and methods.

A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap.

Customized or preset patterns are applied or added to pattern canvases to create new and different pattern images in CorelDRAW. A pattern canvas acts as a background effect to a pattern fill.

{button ,AL(`CLS_PatternCanvases')} **Related Topics**

PatternCanvases.Item

Property **Item**(ByVal **Index** AS Long) AS **PatternCanvas**

PatternCanvases

Description

The **Item** property returns or sets a reference to a specified pattern canvas in the **PatternCanvases** collection in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

In order to reference a specific canvas in the collection, you must pass the index number of a pattern canvas as a parameter:

Parameters	Description
Index	The **Index** parameter is a numerical value that uniquely identifies each pattern canvas within the **PatternCanvases** collection in CorelDRAW. For example, a canvas with an index of 1 is the first pattern canvas in the collection.

Example

The following code example uses the **PutCopy** method to copy data from an existing pattern canvas into a new pattern canvas. The existing pattern canvas is identified by its index number:

```
Sub Canvas()
Dim NewCanvas and New PatternCanvas
NewCanvas.PutCopy PatternCanvases(5)
End Sub
```

{button ,AL(`CLS_PatternCanvases;FNC_Item')} **Related Topics**

PatternCanvases.Count

Property **Count** AS Long

PatternCanvases

Description

The **Count** property returns the number of pattern canvases in the **PatternCanvases** collection in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

The **Count** property returns a Read-Only value.

Example

The following code example removes the third pattern canvas object from the **PatternCanvases** collection and displays the remaining number of canvases in a message box:

```
Sub CanvasCount()
PatternCanvases.Remove (3)
MsgBox .PatternCanvases.Count
End Sub
```

{button ,AL(`CLS_PatternCanvases;FNC_Count')} **Related Topics**

PatternCanvases.Add

Function **Add**(ByRef **PatternCanvas** AS **PatternCanvas**) AS Long

PatternCanvases

Description

The **Add** method adds a pattern canvas to the **PatternCanvases** collection in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

In order to add a specific canvas in the collection, you must pass the **PatternCanvas** value as a parameter:

Parameters	Description
PatternCanvas	The **PatternCanvas** parameter specifies a pattern canvas object in CorelDRAW. This object is added to the **PatternCanvases** collection. Customized or preset patterns are applied or added to pattern canvases to create new and different pattern images in CorelDRAW. A pattern canvas acts as a background effect to a pattern fill.

{button ,AL(`CLS_PatternCanvases;FNC_Add')} **Related Topics**

PatternCanvases.Remove

Sub **Remove**(ByVal **Index** AS Long)

PatternCanvases

Description

The **Remove** method removes a pattern canvas from the **PatternCanvases** collection in CorelDRAW. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW.

In order to remove a specific canvas in the collection, you must pass the index number of a pattern canvas as a parameter:

Parameters	Description
Index	The **Index** parameter is a numerical value that uniquely identifies each pattern canvas within the **PatternCanvases** collection in CorelDRAW. For example, a canvas with an index of 1 is the first pattern canvas in the collection.

Example

The following code example removes the third pattern canvas object from the **PatternCanvases** collection and displays the remaining number of canvases in a message box:

```
Sub CanvasCount()
PatternCanvases.Remove (3)
MsgBox .PatternCanvases.Count
End Sub
```

{button ,AL(`CLS_PatternCanvases;FNC_Remove')} **Related Topics**

Clipboard properties

Clipboard Legend

- Application
- Empty
- Parent
- Valid

Clipboard methods

Clipboard Legend

- Clear
- DataPresent

Clipboard

Class **Clipboard**

Properties Methods Referenced by

The Clipboard class defines the characteristics of Clipboard objects and describes the look and behavior of the objects through its properties and methods.

The clipboard is a temporary storage area that is used to hold cut or copied information. The clipboard is always present in Windows and it stores information until it is replaced by another object or selection that has been cut or copied, or until you quit the current session of Windows.

{button ,AL(`CLS_Clipboard')} **Related Topics**

Clipboard.Application

Property **Application** AS Object

Clipboard

Description

The Application property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The Application property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub VersionApplication()
    MsgBox "You are using CorelDRAW " & Version
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_Application')} **Related Topics**

Clipboard.Parent

Property **Parent** AS Object

Clipboard

Description

The Parent property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The Parent property returns a Read-Only value.

Example

The following code example displays the caption of the clipboard's parent object, the main application window, in a message box. Please note that "vbCrLf" is used for display functionality only:

```
Sub ClipboardParent()
With Clipboard
    MsgBox "The caption of the Clipboard's parent object is: " _
    & vbCrLf & vbCrLf & Parent.AppWindow.Caption
End With
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_Parent')} **Related Topics**

vbCrLf is a VB string constant representing a carriage return-linefeed combination - it has no affect on the functionality of the code.

Clipboard.Valid

Property **Valid** AS Boolean

Clipboard

Description

The Valid property returns a True or False value associated with presence of valid data in the system clipboard. Valid information is anything that can be cut or copied into the clipboard, such as text and graphics selected within a document, or one or more files or folders.

The Valid property returns a Read-Only value.

Example

The following code example checks to see there is valid data in the clipboard. If there is valid data present, it is pasted into the active layer. If there is no valid data in the clipboard, a message displays in a message box:

```
Sub ClipboardValid()
If Clipboard.Valid Then
    ActiveLayer.Paste
Else
    MsgBox "There is no valid data currently in the clipboard."
End If
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_Valid')} **Related Topics**

The clipboard is a temporary storage area that is used to hold cut or copied information. The clipboard is always present in Windows and it stores information until it is replaced by another object or selection that has been cut or copied, or until you quit the current session of Windows.

Clipboard.Empty

Property **Empty** AS Boolean

Clipboard

Description

The Empty property returns a True or False value associated with presence of data in the system clipboard. Unlike the Valid property, which verifies the presence of a specific type or data (i.e. text, graphics), the Empty property looks for the presence of any data in the system clipboard.

The Empty property returns a Read-Only value.

Example

The following code example checks to see if there is any data in the clipboard. If there is data present, it is pasted in the active layer. If there is no data in the system clipboard, a message displays in a message box:

```
Sub ClipboardEmpty()
If Not Clipboard.Empty Then
    ActiveLayer.Paste
Else
    MsgBox "There is no data in the clipboard."
End If
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_Empty')} **Related Topics**

Clipboard.Clear

Sub **Clear**()

Clipboard

Description

The Clear method removes any and all data from the system clipboard. The clipboard should be cleared after large files, no longer needed, are pasted into a document. Clearing the clipboard of these large files frees up any system resources that the file is currently using in the clipboard.

Example

The following code example pastes the current contents of the system clipboard in the active document and uses the Clear method to remove the data from the clipboard::

```
Sub ClipboardClear()
ActiveLayer.Paste
clipboard.Clear
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_Clear')} **Related Topics**

Clipboard.DataPresent

Function **DataPresent**(ByVal **FormatName** AS String) AS Boolean

Clipboard

Description

The DataPresent method returns a True or False value that verifies the presence of a specific data format in the system clipboard. The data format (e.g. text, graphic, etc…) is determined by the Windows Clipboard Viewer utility. If you click the Display menu in the Viewer, all available data formats are available, with a check mark next to the active format. You may select other formats to change the current view of the data in the clipboard. The clipboard may contain more than one data format at a time.

Parameters	Description
FormatName	FormatName is a string parameter that is passed to the DataFormat function to determine presence of a particular data format in the clipboard. It must be contained in quotes when passed to the function. "Text" is an example of a string parameter that may be passed as a FormatName parameter.

Example

The following code example verifies the presence of text data in the clipboard. It text data is present, the clipboard is cleared of any data with the Clear method. If there is no text data in the clipboard, a message displays in a message box:

```
Sub PresentData()
If clipboard.DataPresent("Text") Then
    clipboard.Clear
Else
    MsgBox "There is no text data in the clipboard."
End If
End Sub
```

{button ,AL(`CLS_Clipboard;FNC_DataPresent')} **Related Topics**

A data type consisting of a sequence of contiguous characters that represent the characters themselves rather than their numeric values. A String can include letters, numbers, spaces, and punctuation. The String data type can store fixed-length strings ranging in length from 0 to approximately 63K characters and dynamic strings ranging in length from 0 to approximately 2 billion characters. The dollar sign ($) type-declaration character represents a String in Visual Basic.

Rulers properties

Rulers Legend

- Application
  - HUnits
- Parent
  - VUnits

Rulers

Class **Rulers**

Properties Referenced by

The **Rulers** class defines the characteristics of ruler collection objects and describes the look and behavior of the objects through its properties and methods.

A ruler is a measuring tool that displays on the left side and along the top of the Application Window. The rulers help you size and position the objects in your drawing.

You can choose when and how you want to display the rulers and the grid. If screen space is limited, you might choose to hide the rulers and display them only when you need them. Or, if you want to view your drawing as it will appear when you print it, you can hide the grid and display it later. The rulers and grid maintain their settings even when you hide them.

The rulers can help you determine the size and position of objects. Before using the rulers, determine the position of the ruler origin - the place where the rulers' 0 points intersect. Putting the ruler origin exactly where you want it on the Drawing Page ensures that the ruler coordinates begin from the exact location you need.

In addition to positioning the ruler origin, you can move the rulers within the Drawing Window so that you can use them most effectively. For example, you can move the rulers over your drawing to create or move an object with precision.

You can change the units of measurement displayed on the Horizontal and Vertical rulers. CorelDRAW provides an array of units, ranging from points, millimeters, and inches to larger units such as meters, kilometers, and miles. Use the unit setting that best suits the type and size of drawing you want to create.

{button ,AL(`CLS_Rulers')} **Related Topics**

Rulers.Application

Property **Application** AS **Application**

Rulers

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub RulersApp()
With ActiveDocument.Rulers
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Rulers;FNC_Application')} **Related Topics**

Rulers.Parent

Property **Parent** AS **Document**

Rulers

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the ruler's parent active page in CorelDRAW:

```
Sub RulersParent()
With ActiveDocument.Rulers
    MsgBox .Parent.ActivePage.Name
End With
End Sub
```

{button ,AL(`CLS_Rulers;FNC_Parent')} **Related Topics**

Rulers.VUnits

Property **VUnits** AS **cdrUnit**

Rulers

Description

The **VUnits** property returns or sets a value associated with the unit of measurement in the vertical rulers of CorelDRAW. A ruler is a measuring tool that displays on the left side and along the top of the Application Window. The rulers help you size and position the objects in your drawing.

The **VUnits** property returns or sets a value of cdrUnit.

Example

The following code example sets the unit of measurement for the vertical rulers to inches in CorelDRAW:

```
Sub RulersVertical()
With ActiveDocument.Rulers
    .VUnits = cdrInch
End With
```

{button ,AL(`CLS_Rulers;FNC_VUnits')} **Related Topics**

Rulers.HUnits

Property **HUnits** AS **cdrUnit**

Rulers

Description

The **HUnits** property returns or sets a value associated with the unit of measurement in the horizontal rulers of CorelDRAW. A ruler is a measuring tool that displays on the left side and along the top of the Application Window. The rulers help you size and position the objects in your drawing.

The **HUnits** property returns or sets a value of cdrUnit.

Example

The following code example sets the unit of measurement for the horizontal rulers to inches in CorelDRAW:

```
Sub RulersHorizontal()
With ActiveDocument.Rulers
    .HUnits = cdrInch
End With
```

{button ,AL(`CLS_Rulers;FNC_HUnits')} **Related Topics**

Grid properties

Grid Legend

- Application

- Parent
  - Snap
  - SpacingX
  - SpacingY
  - Type
  - Visible

Grid methods

Grid Legend

SetFrequency

Grid

Class **Grid**

Properties Methods Referenced by

The **Grid** class defines the characteristics of Grid objects and describes the look and behavior of the objects through its properties and methods. A Grid is a series of evenly spaced horizontal and vertical dots that are used to help draw and arrange objects. You can use the controls on the Grid and Ruler Setup dialog box to set the grid's parameters. For greater accuracy, you can also have objects in your illustration snap to the grid when you move, or draw them.

You can set the spacing between grid lines using two options: frequency and spacing. Frequency sets the distance between grid lines according to how many grid lines you want per horizontal and vertical unit. Spacing sets this distance by specifying the exact distance you want between each line. The grid acts the same way no matter which option you choose.

By default, CorelDRAW displays the grid as lines resembling graph paper. You can choose to display the grid as dots, which are not as noticeable. The points where horizontal and vertical lines intersect represent grid dots. The frequency or spacing settings you specify apply to both lines and dots.

A grid object must be ungrouped before you can make changes to its shape. Once you ungroup the grid object, it becomes a set of individual rectangles.The Snap To Grid feature helps you align objects precisely. When you enable Snap To Grid, objects you move or draw automatically snap to the grid so that they line up vertically and horizontally with the nearest grid marker.

You can choose when and how you want to display the rulers and the grid. The rulers and grid maintain their settings even when you hide them.

{button ,AL(`CLS_Grid')} **Related Topics**

Grid.Application

Property **Application** AS **Application**

Grid

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub GridApp()
With ActiveDocument.Grid
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_Application')} **Related Topics**

Grid.Parent

Property **Parent** AS **Document**

Grid

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the grid object's parent layer in a message box:

```
Sub GridParent()
With ActiveDocument.Grid
    MsgBox .Parent.ActiveLayer.Name
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_Parent')} **Related Topics**

Grid.Visible

Property **Visible** AS Boolean

Grid

Description

The **Visible** property returns or sets a True or False value indicating the visibility status of a grid object in CorelDRAW. A Grid is a series of evenly spaced horizontal and vertical dots that are used to help draw and arrange objects. You can use the controls on the Grid and Ruler Setup dialog box to set the grid's parameters. For greater accuracy, you can also have objects in your illustration snap to the grid when you move, or draw them.

If the **Visible** property is True, the grid appears in the Drawing Window of CorelDRAW. This property allows you to show or hide the grid object.

Example

The following code example sets the **Visible** property of the grid to True, displaying the grid in the Drawing Window. The grid type pattern is set to dots and the **SetFrequency** method places 5 horizontal and vertical grid dots per square inch in the grid:

```
Sub GridVisible()
With ActiveDocument.Grid
    .Visible = True
    .Type = cdrGridDot
    .SetFrequency 5, 5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_Visible')} **Related Topics**

cdrGridDot=0

cdrGridLine=1

Grid.Type

Property **Type** AS **cdrGridType**

Grid

Description

The **Type** property returns or sets a value indicating the grid type in CorelDRAW. A Grid is a series of evenly spaced horizontal and vertical dots that are used to help draw and arrange objects. You can use the controls on the Grid and Ruler Setup dialog box to set the grid's parameters. For greater accuracy, you can also have objects in your illustration snap to the grid when you move, or draw them.

By default, CorelDRAW displays the grid as lines resembling graph paper. You can choose to display the grid as dots, which are not as noticeable. The points where horizontal and vertical lines intersect represent grid dots. The frequency or spacing settings you specify apply to both lines and dots.

The **Type** property returns a value of cdrGridType.

Example

The following code example sets the **Visible** property of the grid to True, displaying the grid in the Drawing Window. The grid type pattern is set to dots and the **SetFrequency** method places 5 horizontal and vertical grid dots per square inch in the grid:

```
Sub GridType()
With ActiveDocument.Grid
    .Visible = True
    .Type = cdrGridDot
    .SetFrequency 5, 5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_Type')} **Related Topics**

Grid.Snap

Property **Snap** AS Boolean

Grid

Description

The **Snap** property returns a True or False value indicating if a grid's snap capability is enabled in CorelDRAW. A grid's ability to snap objects means it can force an object that is being drawn or moved to align automatically to a point on the grid, a guideline, or another object.

A Grid is a series of evenly spaced horizontal and vertical dots that are used to help draw and arrange objects. You can use the controls on the Grid and Ruler Setup dialog box to set the grid's parameters. For greater accuracy, you can also have objects in your illustration snap to the grid when you move, or draw them.

The Snap To Grid feature helps you align objects precisely. When you enable Snap To Grid, objects you move or draw automatically snap to the grid so that they line up vertically and horizontally with the nearest grid marker. If the Snap property is set to True, the Snap to Grid feature is enabled in CorelDRAW.

Example

The following code example sets the **Snap** property of the grid to True, sets the grid type pattern to line, and sets the spacing within the grid. By setting the **SpacingX** and **SpacingY** values to .05, a vertical and horizontal grid line appears every .5 inches in the grid:

```
Sub GridSnap()
With ActiveDocument.Grid
    .Snap = True
    .Type = cdrGridLine
    .SpacingX = .5
    .SpacingY = .5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_Snap')} **Related Topics**

Grid.SetFrequency

Sub **SetFrequency**(ByVal **GridX** AS Double, ByVal **GridY** AS Double)

Grid

Description

The **SetFrequency** method determines how many grid lines or dots appear per square inch in a CorelDRAW grid. A grid's ability to snap objects means it can force an object that is being drawn or moved to align automatically to a point on the grid, a guideline, or another object.

In order to define the grid frequency, you must past the **GridX** and **GridY** values as parameters:

Parameters	Description
GridX	The **GridX** parameter is a numerical value that determines now many horizontal grid lines or dotesappear per inch in a grid. If the GridX value is 4, there are 4 horizontal grid lines or dotes per inch in the grid. The minimum value is .005.
GridY	The **GridY** parameter is a numerical value that determines now many vertical grid lines or dotes appear per inch in a grid. If the GridX value is 4, there are 4 vertical grid lines or dots per inch in the grid. The minimum value is .005.

Example

The following code example sets the **Visible** property of the grid to True, displaying the grid in the Drawing Window. The grid type pattern is set to dots and the **SetFrequency** method places 5 horizontal and vertical grid dots per square inch in the grid:

```
Sub GridFreq()
With ActiveDocument.Grid
    .Visible = True
    .Type = cdrGridDot
    .SetFrequency 5, 5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_SetFrequency')} **Related Topics**

Grid.SpacingX

Property **SpacingX** AS Double

Grid

Description

The **SpacingX** property returns or sets a value associated with the horizontal spacing between grid lines or dots in a CorelDRAW grid. A grid's ability to snap objects means it can force an object that is being drawn or moved to align automatically to a point on the grid, a guideline, or another object.

The **SpacingX** property defines how often a horizontal grid line or dot appears in the grid, in a measurement of inches. For example, `Grid.SpacingX = .5` places a horizontal grid line or dot every .5 inches throughout the grid. The minimum value is .0001 horizontal lines or dots per inch.

Example

The following code example sets the **Snap** property of the grid to True, sets the grid type pattern to line, and sets the spacing within the grid. By setting the **SpacingX** and **SpacingY** values to .05, a vertical and horizontal grid line appears every .5 inches in the grid:

```
Sub GridSpacingX()
With ActiveDocument.Grid
    .Snap = True
    .Type = cdrGridLine
    .SpacingX = .5
    .SpacingY = .5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_SpacingX')} **Related Topics**

Grid.SpacingY

Property **SpacingY** AS Double

Grid

Description

The **SpacingY** property returns or sets a value associated with the horizontal spacing between grid lines or dots in a CorelDRAW grid. A grid's ability to snap objects means it can force an object that is being drawn or moved to align automatically to a point on the grid, a guideline, or another object.

The **SpacingY** property defines how often a vertical grid line or dot appears in the grid, in a measurement of inches. For example, `Grid.SpacingY = .5` places a vertical grid line or dot every .5 inches throughout the grid. The minimum value is .0001 vertical lines or dots per inch.

Example

The following code example sets the **Snap** property of the grid to True, sets the grid type pattern to line, and sets the spacing within the grid. By setting the **SpacingX** and **SpacingY** values to .05, a vertical and horizontal grid line appears every .5 inches in the grid:

```
Sub GridSpacingY()
With ActiveDocument.Grid
    .Snap = True
    .Type = cdrGridLine
    .SpacingX = .5
    .SpacingY = .5
End With
End Sub
```

{button ,AL(`CLS_Grid;FNC_SpacingY')} **Related Topics**

PowerClip properties

PowerClip Legend

- Application
  ContentsLocked
- Parent
- Shapes

PowerClip methods

PowerClip Legend

- EnterEditMode
- ExtractShapes
- LeaveEditMode

PowerClip

Class **PowerClip**

Properties Methods Referenced by

The **PowerClip** class defines the characteristics of powerclip objects and describes the look and behavior of the collection objects through its properties and methods.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

When you PowerClip objects, you put one object inside another object or group of objects. One object becomes the contents, while the other becomes the container. You can create a container from any object you create using CorelDRAW, including shapes, lines, curves, Artistic text, and groups. A contents object can be any object you create using CorelDRAW or import from another program.

When you're creating PowerClip object, you may want to compare the container object to a window. Just as a window's frame represents the limits of what you can see, a container object lets you see only the portion of a contents object (or group of objects) that fits inside the container's boundaries. If the size of the contents object exceeds that of its container, CorelDRAW automatically crops the contents object. You see only the portion of the contents object that fits inside the container.

By default, CorelDRAW automatically centers PowerClip contents objects inside their containers. However, you can change the default so that contents objects are offset from the center of the container.

After you create a PowerClip object, you can edit it to change the way the contents and container objects interact.

When you edit the contents object, CorelDRAW temporarily separates the contents and container objects. You can then edit the contents of the PowerClip object, for example, by changing its fill and outline properties, without altering the container object. When you lock or unlock the contents of a PowerClip object, you control the interaction between the contents and container objects. When a contents object is locked to its container object, both objects undergo the same changes. For example, when a PowerClip object is moved, rotated, or resized, both the contents and container objects are affected in the same way. When the contents object is unlocked, it's locked to the page and remains stationary even if you move or rotate its container.

By extracting the contents from a PowerClip object, the container and contents objects return to being separate objects.

{button ,AL(`CLS_PowerClip')} **Related Topics**

PowerClip.ExtractShapes

Function **ExtractShapes**() As **ShapeRange**

Description

The **ExtractShapes** method removes a powerclip shape from its container in CorelDRAW. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

By extracting the contents from a PowerClip object, the container and contents objects return to being separate objects.

Example

The following code example unlocks the contents of the active shape's powerclip object, counts the number of shapes contained in the powerclip, and extracts the shape objects. The code leaves edit mode when the shapes have been extracted, and the contents of the powerclip are locked:

```
Sub PowerClipEdit()
With ActiveShape.PowerClip
    .ContentsLocked = False
    .EnterEditMode
    .ExtractShapes
    .LeaveEditMode
    .ContentsLocked = True
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_ExtractShapes')} **Related Topics**

PowerClip.Application

Property **Application** AS **Application**

PowerClip

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub PowerClipApp()
With ActiveShape.PowerClip
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_Application')} **Related Topics**

PowerClip.Parent

Property **Parent** AS **Shape**

PowerClip

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the power clip object's parent layer in a message box:

```
Sub Layer()
MsgBox ActiveShape.PowerClip.Parent.Layer.Name
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_Parent')} **Related Topics**

PowerClip.Shapes

Property **Shapes** AS **Shapes**

PowerClip

Description

The **Shapes** property returns the shape objects that are used in a powerclip in CorelDRAW. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

The **Shapes** property returns a Read-Only value.

Example

The following code example counts the number of shape objects that are in the powerclip object of the active shape:

```
Sub PowerClipCount()
With ActiveShape.PowerClip
    MsgBox .Shapes.Count
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_Shapes')} **Related Topics**

PowerClip.ContentsLocked

Property **ContentsLocked** AS Boolean

PowerClip

Description

The **ContentsLocked** property returns or sets a True or False value that indicates whether or not the contents of a powerclip are locked in CorelDRAW. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

If the **ContentsLocked** property is True, the powerclip contents are locked.

When you lock or unlock the contents of a PowerClip object, you control the interaction between the contents and container objects. When a contents object is locked to its container object, both objects undergo the same changes. For example, when a PowerClip object is moved, rotated, or resized, both the contents and container objects are affected in the same way. When the contents object is unlocked, it's locked to the page and remains stationary even if you move or rotate its container.

Example

The following code example unlocks the contents of the active shape's powerclip object, counts the number of shapes contained in the powerclip, and extracts the shape objects. The code leaves edit mode when the shapes have been extracted, and the contents of the powerclip are locked:

```
Sub PowerClipEdit()
With ActiveShape.PowerClip
    .ContentsLocked = False
    .EnterEditMode
    .ExtractShapes
    .LeaveEditMode
    .ContentsLocked = True
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_ContentsLocked')} **Related Topics**

PowerClip.EnterEditMode

Sub **EnterEditMode**()

PowerClip

Description

The **EnterEditMode** method enables editing of a powerclip in CorelDRAW by entering edit mode. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

When you edit the contents object, CorelDRAW temporarily separates the contents and container objects. You can then edit the contents of the PowerClip object, for example, by changing its fill and outline properties, without altering the container object.

Example

The following code example unlocks the contents of the active shape's powerclip object, counts the number of shapes contained in the powerclip, and extracts the shape objects. The code leaves edit mode when the shapes have been extracted, and the contents of the powerclip are locked:

```
Sub PowerClipEdit()
With ActiveShape.PowerClip
    .ContentsLocked = False
    .EnterEditMode
    .ExtractShapes
    .LeaveEditMode
    .ContentsLocked = True
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_EnterEditMode')} **Related Topics**

PowerClip.LeaveEditMode

Sub **LeaveEditMode**()

PowerClip

Description

The **LeaveEditMode** method disables editing of a powerclip in CorelDRAW by leaving edit mode. Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

When you edit the contents object, CorelDRAW temporarily separates the contents and container objects. You can then edit the contents of the PowerClip object, for example, by changing its fill and outline properties, without altering the container object.

Example

The following code example unlocks the contents of the active shape's powerclip object, counts the number of shapes contained in the powerclip, and extracts the shape objects. The code leaves edit mode when the shapes have been extracted, and the contents of the powerclip are locked:

```
Sub PowerClipEdit()
With ActiveShape.PowerClip
    .ContentsLocked = False
    .EnterEditMode
    .ExtractShapes
    .LeaveEditMode
    .ContentsLocked = True
End With
End Sub
```

{button ,AL(`CLS_PowerClip;FNC_LeaveEditMode')} **Related Topics**

DataItems properties

DataItems Legend

- Application
- Count
-
- Item
- Parent

DataItems methods

DataItems Legend

- Add
- Clear
- CopyFrom

DataItems

Class **DataItems**

Properties Methods Referenced by

The **DataItems** class defines the characteristics of data item collection objects and describes the look and behavior of the objects through its properties and methods. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

You can change the name of any field to better suit your object data summary. You can also change the location of fields to make them appear in a logical order on the data summary.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

You can create and assign as many data fields as you want, as long as they use permitted format variables.

You can delete any data field except for CDRStaticID. When you delete a field, you also delete all data entered for that field in the active document.

{button ,AL(`CLS_DataItems')} **Related Topics**

DataItems.Application

Property **Application** AS **Application**

DataItems

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub AppName()
With ActiveShape.ObjectData
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Application')} **Related Topics**

DataItems.Parent

Property **Parent** AS **Shape**

DataItems

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the data item collection's parent object in a message box:

```
Sub FieldName()
With ActiveShape.ObjectData
    MsgBox .Parent.Name
End With
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Parent')} **Related Topics**

DataItems.Count

Property **Count** AS Long

DataItems

Description

The **Count** property returns the number of fields in a data item object in the **DataItems** collection of CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

This property returns a Read-Only value.

Example

The following code example counts the number of data fields in the active shape of CorelDRAW:

```
Sub Test()
    Dim n As Long
    CountFields = ActiveSelection.Shapes(1).ObjectData.Count
    MsgBox CountFields
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Count')} **Related Topics**

DataItems.Item

Property **Item**(ByVal **IndexOrName** AS Variant) AS **DataItem**

DataItems

Description

The **Item** property returns a string value associated with the index number of a data item object in the **DataItems** collection of CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

`ObjectData(2)` refers to the second data item object in the **DataItems** collection of CorelDRAW.

The **Item** property returns a Read-Only value.

The **Item** property is the default property and may be omitted when referencing items in the collection. For example, `ObjectData.Item(2)` is the same as `ObjectData(2)` - they both reference the second data item object in the **DataItems** collection.

You must reference a data item object in the collection by passing its index number as a parameter:

Parameters	Description
IndexOrName	**Index** is a preset placeholder for each object in a **DataItems** collection; it uniquely identifies each member of the collection. **Name** is the unique text name given to each data item.

Example

The following code example displays the name of a data field in an object data item in CorelDRAW:

```
Sub FieldName()
With ActiveShape.ObjectData.Item(1)
    MsgBox .DataField.Name
End With
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Item')} **Related Topics**

DataItems.Add

Function **Add**(ByVal **DataField** As **DataField**, [ByVal Value As Variant]) As **DataItem**

Description

The **Add** method adds a data field to a data item object in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

A data field is a single property or value associated with a data item object.

Parameters	Description
DataField	This parameter identifies the data field that is added to a data object in CorelDRAW. A data field is a single property or value associated with a data item object.
Value	This parameter associated a value with the new data field. This value is optional.

Example

The following code example adds a new data field with name "Weight" and specifies custom data format (2 digits after decimal point and "kg" suffix after the number). Then sets the value of 24 to the current object:

```
Sub Test()
    Dim s As Shape
    Dim di As DataItem
    Set s = ActiveSelection.Shapes(1)
    Set di = s.ObjectData.Add("Weight", "#,##0.00 " + Chr$(34) + "kg" + Chr$(34))
    di.Value = 24
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Add')} **Related Topics**

DataItems.Clear

Sub **Clear**()

Description

The **Clear** method clears the values of the entire **DataItems** collection in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

Example

The following code example clears the value from all data fields in the data items collection of CorelDRAW:

```
Sub ClearValue()
With ActiveShape.ObjectData
    .Clear
End With
End Sub
```

{button ,AL(`CLS_DataItems;FNC_Clear')} **Related Topics**

DataItems.CopyFrom

Sub **CopyFrom**(ByVal **Shape** As **Shape**)

Description

The **CopyFrom** method copies the data fields and values from a data item object and copies them into another data item object in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

Parameters	Description
Shape	The **Shape** parameter identifies the shape object who's data fields and values are copied and placed into another data item object in CorelDRAW.

{button ,AL(`CLS_DataItems;FNC_CopyFrom')} **Related Topics**

DataItem properties

DataItem Legend

- Application
- DataField
- FormattedValue
- Parent
- Value

DataItem methods

DataItem Legend

Clear

DataItem

Class **DataItem**

Properties Methods Referenced by

The **DataItem** class defines the characteristics of data item objects and describes the look and behavior of the objects through its properties and methods. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

If the preset formats in CorelDRAW don't provide the information you want in your data summary, you can create your own custom formats using the variables available for the format type you're using. The field format you select is used for all objects in the active drawing.

You can change any object data field by giving it a preset or custom format. For example, you can change a numeric field to display more or fewer decimal places, or to display numbers in thousands.

You can change the name of any field to better suit your object data summary. You can also change the location of fields to make them appear in a logical order on the data summary.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

You can create and assign as many data fields as you want, as long as they use permitted format variables.

You can delete any data field except for CDRStaticID. When you delete a field, you also delete all data entered for that field in the active document.

{button ,AL(`CLS_DataItem')} **Related Topics**

DataItem.DataField

Property **DataField** As **DataField**

Description

The **DataField** property returns a value associated with a data field in an data item object in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

You can create and assign as many data fields as you want, as long as they use permitted format variables.

You can delete any data field except for CDRStaticID. When you delete a field, you also delete all data entered for that field in the active document.

The **DataField** property returns a Read-Only value.

Example

The following code example displays the name of a data field in an object data item in CorelDRAW:

```
Sub FieldName()
With ActiveShape.ObjectData.Item(1)
    MsgBox .DataField.Name
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_DataField')} **Related Topics**

DataItem.Application

Property **Application** AS **Application**

DataItem

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub AppName()
With ActiveShape.ObjectData.Item(1)
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_Application')} **Related Topics**

DataItem.Parent

Property **Parent** AS **DataItems**

DataItem

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the number of data item objects in the **DataItems** collection of CorelDRAW:

```
Sub FieldName()
With ActiveShape.ObjectData.Item(1)
    MsgBox .Parent.Count
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_Parent')} **Related Topics**

DataItem.Value

Property **Value** AS Variant

DataItem

Description

The **Value** property returns or sets the value of a data item object in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

Value is a default property of the DataItem object returning the value associated with the shape.

Example

The following code example displays the value associated with a data item object in CorelDRAW:

```
Sub ItemValue()
With ActiveShape.ObjectData.Item(1)
    MsgBox .Value
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_Value')} **Related Topics**

DataItem.Clear

Sub **Clear**()

Description

The **Clear** method removes the value from a data field in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

The **Clear** method removes the value - it doesn't not remove the data field from the data item object from CorelDRAW.

Example

The following code example clears the value of a data field object in CorelDRAW:

```
Sub ClearValue()
With ActiveShape.ObjectData.Item(1)
    .Clear
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_Clear')} **Related Topics**

DataItem.FormattedValue

Property **FormattedValue** AS String

DataItem

Description

The **FormattedValue** property returns a value associated with a formatted data field in CorelDRAW. A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

If you require custom fields, you can define their formats using the controls in the Format Definition dialog box. This dialog box gives you access to four basic field formats: General, Date/Time, Linear/Angular, and Numeric. Each of these formats provides a series of common settings. You can use these formats or create your own.

The **FormattedValue** returns a Read-Only value.

Example

The following code example displays the value of a data field after it has been formatted in CorelDRAW:

```
Sub FieldFormat()
With ActiveShape.ObjectData.Item(1)
    MsgBox .FormattedValue
End With
End Sub
```

{button ,AL(`CLS_DataItem;FNC_FormattedValue')} **Related Topics**

Transparency properties

Transparency Legend

- Application
  - AppliedTo
  - End
  - Fountain
- Frozen
- Parent
  - Pattern
  - Start
  - Texture
- Type
  - Uniform

Transparency methods

Transparency Legend

- ApplyFountainTransparency
- ApplyNoTransparency
- ApplyPatternTransparency
- ApplyTextureTransparency
- ApplyUniformTransparency
- Freeze
- Unfreeze

Transparency

Class **Transparency**

Properties Methods Referenced by

The **Transparency** class defines the characteristics of transparency objects and describes the look and behavior of the objects through its properties and methods. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

When you apply a transparency to an object, you're creating a grayscale mask using the uniform, fountain, texture, or pattern fill type, which is applied on top of the object. By positioning a transparent object on top of another object, you simulate a lens. Since a transparency is applied on top of an object, the object's color shows through the transparency. You can use fountain, uniform, texture, and pattern fill types to create your transparency.

Additionally, you can edit transparencies. For example, you can change the fill type or change the level of opacity, which is determined by the grayscale level you specify.

A Uniform transparency is filled with a solid color. The color is consistent throughout the object to which it's applied. A Fountain transparency flows smoothly from one color to another. The transparency can flow in a straight line across the object (linear), in concentric circles from the center of the object out (radial), in rays from the center of the object out (conical), or in concentric squares from the center of the object out (square). A Texture transparency is a random, fractally generated transparency that you can use to give your objects a natural appearance. Transparency handles let you control the block of fractal texture that controls the transparency of the object. A Pattern transparency is a pre-generated, symmetrical image that is repeated over and over, making it extremely useful for creating tiles. You can fill an object completely with one image, but you would more often use a series of repeated images to form a tiled fill. The effect is similar to applying wallpaper to a wall.

{button ,AL(`CLS_Transparency')} **Related Topics**

Transparency.AppliedTo

Property **AppliedTo** As **cdrTransparencyAppliedTo**

Description

The **AppliedTo** property returns or sets the value associated with the application of a transparency in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image. A transparency can be uniform, pattern, fountain or texture types. An object may also have no transparency.

A transparency can be applied to an object's fill, outline, or both the fill and outline.

Example

The following code example displays the type of object that the transparency is applied to in the active shape of CorelDRAW:

```
Sub Trans()
With ActiveShape.Transparency
    MsgBox .AppliedTo
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_AppliedTo')} **Related Topics**

Transparency.Application

Property **Application** AS Object

Transparency

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub App()
MsgBox ActiveShape.Transparency.Application.Version
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Application')} **Related Topics**

Transparency.Parent

Property **Parent** AS Object

Transparency

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

{button ,AL(`CLS_Transparency;FNC_Parent')} **Related Topics**

cdrNoTransparency=0

cdrUniformTransparency=1

cdrFountainTransparency=2

cdrPatternTransparency=3

cdrTextureTransparency=4

Transparency.Type

Property **Type** AS **cdrTransparencyType**

Transparency

Description

The **Type** property returns or sets the transparency type of an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image. A transparency can be uniform, pattern, fountain or texture types. An object may also have no transparency.

The **Type** property returns cdrTransparencyType.

Example

The following code example displays a value associated with the transparency type in the active shape of CorelDRAW:

```
Sub Trans()
MsgBox ActiveShape.Transparency.Type
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Type')} **Related Topics**

Transparency.Uniform

Property **Uniform** AS Long

Transparency

Description

The **Uniform** property returns or sets a uniform transparency in an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A uniform transparency is filled with a solid color. The color is consistent throughout the object to which it's applied.

Example

The following code example applies a transparency to the active shape of CorelDRAW, with a transparency level of 20. Using the **Uniform** property, the transparency level is increased to 90, which makes the active shape more transparent:

```
Sub Trans()
With ActiveShape.Transparency
    .ApplyUniformTransparency 20
    .Uniform = 90
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Uniform')} **Related Topics**

Transparency.Fountain

Property **Fountain** AS **FountainFill**

Transparency

Description

The **Fountain** property returns or sets a fountain transparency in an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Fountain transparency flows smoothly from one color to another. The transparency can flow in a straight line across the object (linear), in concentric circles from the center of the object out (radial), in rays from the center of the object out (conical), or in concentric squares from the center of the object out (square).

Example

The following code example selects an object in the active document and determines if the object is a curve object. If the selection is a curve, it changes its **Closed** status to True, joining all the segments in the shape to make a self-contained object. A color fountain transparency is applied to the new closed curve object and using the Fountain property, the angle in the fountain transparency effect is set to 45 degrees:

```
Sub CurveClosed()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    s.Curve.Closed = True
End If
s.Transparency.ApplyFountainTransparency CreateColorEx(5005, 255, 0, 0), _
CreateColorEx(5005, 0, 0, 0), 1
s.Transparency.Fountain.Angle = 45
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Fountain')} **Related Topics**

Transparency.Pattern

Property **Pattern** AS **PatternFill**

Transparency

Description

The **Pattern** property returns or sets a pattern transparency in an object in CorelDRAW Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Pattern transparency is a pre-generated, symmetrical image that is repeated over and over, making it extremely useful for creating tiles. You can fill an object completely with one image, but you would more often use a series of repeated images to form a tiled fill. The effect is similar to applying wallpaper to a wall.

Example

The following code example applies a two color "checkers" pattern fill to the active shape in CorelDRAW. The fill will transform its shape when the object's shape is altered. The path file of the pattern is displayed in a message box, and the frozen status of the transparency displays as well::

```
Sub FillPattern()
Dim ColorFront as Color
Dim ColorBack as Color
Set ColorFront = ActivePalette.Colors(3)
Set ColorBack = ActivePalette.Colors(4)
With ActiveShape.Transparency
.ApplyPatternTransparency cdrTwoColorPattern, "checkers", 1, ColorFront, ColorBack, True
    MsgBox .Pattern.FilePath
    MsgBox .Frozen
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Pattern')} **Related Topics**

Transparency.Texture

Property **Texture** AS **TextureFill**

Transparency

Description

The **Texture** property returns or sets a texture transparency in an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Texture transparency is a random, fractally generated transparency that you can use to give your objects a natural appearance. Transparency handles let you control the block of fractal texture that controls the transparency of the object.

Example

The following code example unfreezes the transparency in the active shape and applies a texture transparency. The texture pattern is called "Air Brush" and it is found in the "Samples" texture library. The **Texture** property is set up to transform the texture transparency as the shape is altered, and the transparency is frozen to prevent changes being made to the active shape.

```
Sub TransTexture()
With ActiveShape.Transparency
    .Unfreeze
    .ApplyTextureTransparency "Air Brush", "Samples"
    .Texture.TransformWithShape = True
    .Freeze
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Texture')} **Related Topics**

Transparency.Start

Property **Start** AS Long

Transparency

Description

The **Start** property returns or sets a percentage value that indicates the level of transparency in the beginning of a fountain transparency. Higher values indicate a greater degree of transparency. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

Example

The following code example sets the transparency level for the beginning and the end of a transparency effect in the active shape of CorelDRAW. Both values are set to 100%, which means both ends of the transparency are completely transparent:

```
Sub Trans()
With ActiveShape.Transparency
    .Start = 100
    .End  = 100
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Start')} **Related Topics**

Transparency.End

Property **End** AS Long

Transparency

Description

The **End** property returns or sets a percentage value that indicates the level of transparency at the end of a fountain transparency. Higher values indicate a greater degree of transparency. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

Example

The following code example sets the transparency level for the beginning and the end of a transparency effect in the active shape of CorelDRAW. Both values are set to 100%, which means both ends of the transparency are completely transparent:

```
Sub Trans()
With ActiveShape.Transparency
    .Start = 100
    .End  = 100
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_End')} **Related Topics**

Transparency.Frozen

Property **Frozen** AS Boolean

Transparency

Description

The **Frozen** property returns a True or False value that indicates if a transparency is frozen in CorelDRAW. You can freeze the contents of a transparency to create a bitmap. Once frozen, the contents of the object no longer interact with other objects on the screen. For example, when you move the frozen transparency, its contents don't change.

Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

The **Frozen** property returns a Read-Only value.

Example

The following code example applies a two color "checkers" pattern fill to the active shape in CorelDRAW. The fill will transform its shape when the object's shape is altered. The path file of the pattern is displayed in a message box, and the frozen status of the transparency displays as well::

```
Sub FillPattern()
Dim ColorFront as Color
Dim ColorBack as Color
Set ColorFront = ActivePalette.Colors(3)
Set ColorBack = ActivePalette.Colors(4)
With ActiveShape.Transparency
.ApplyPatternTransparency cdrTwoColorPattern, "checkers", 1, ColorFront, ColorBack, True
    MsgBox .Pattern.FilePath
    MsgBox .Frozen
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Frozen')} **Related Topics**

Transparency.ApplyNoTransparency

Sub **ApplyNoTransparency**()

Transparency

Description

The **ApplyNoTransparency** method removes all transparencies from an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

Example

The following code example clears the transparency effect from the active shape of CorelDRAW, if a transparency has been applied to the shape object:

```
Sub Trans()
With ActiveShape.Transparency
    .ApplyNoTransparency
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_ApplyNoTransparency')} **Related Topics**

Transparency.ApplyUniformTransparency

Sub **ApplyUniformTransparency**(ByVal **Value** AS Long)

Transparency

Description

The **ApplyUniformTransparency** method applies a uniform fill transparency to an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Uniform transparency is filled with a solid color. The color is consistent throughout the object to which it's applied.

In order to use the **ApplyUniformTransparency** method, you must pass the **Value** parameter:

Parameters	Description
Value	The **Value** parameter defines the level of transparency in the uniform transparency fill. The Value parameter is a percentage from 0 to 100. The higher the value, the more transparent the uniform fill transparency in an object.

Example

The following code example applies a transparency to the active shape of CorelDRAW, with a transparency level of 20. Using the **Uniform** property, the transparency level is increased to 90, which makes the active shape more transparent:

```
Sub Trans()
With ActiveShape.Transparency
    .ApplyUniformTransparency 20
    .Uniform = 90
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_ApplyUniformTransparency')} **Related Topics**

Transparency.ApplyFountainTransparency

Function **ApplyFountainTransparency**(ByVal Start AS Long, ByVal End AS Long, ByVal Type AS **cdrFountainFillType**, ByVal Angle AS Double, ByVal Steps AS Long, ByVal EdgePad AS Long, ByVal MidPoint AS Long) AS **FountainFill**

Transparency

Description

The **ApplyFountainTransparency** method applies a fountain fill transparency to an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Fountain transparency flows smoothly from one color to another. The transparency can flow in a straight line across the object (linear), in concentric circles from the center of the object out (radial), in rays from the center of the object out (conical), or in concentric squares from the center of the object out (square).

In order to use the **ApplyFountainTransparency** method, you must pass several values as parameters:

Parameters	Description
Start	The **Start** parameter is a percentage value that indicates the level of transparency in the beginning of a fountain transparency. Higher values indicate a greater degree of transparency. This value is optional and the default value is 0.
End	The **End** parameter is a percentage value that indicates the level of transparency in the end of a fountain transparency. Higher values indicate a greater degree of transparency. This value is optional and the default value is 100.
Type	The **Type** parameter defines the fountain fill type in the transparency effect. A fountain fill can be linear, conical, radial, or square. The Type parameter returns cdrFountainFillType. This value is optional and the default value is cdrLinearFountainFill (1).
Angle	The **Angle** parameter defines the degree of the angle in linear, conical or square fountain fill transparencies. Changing the angle of gradation affects the slant of the fountain fill. Radial fountain fills progress in a series of concentric circles, so you can't change their angle. Positive values rotate the fill counterclockwise; negative values rotate it clockwise. This value is optional and the default value is 0.
Steps	The **Steps** parameter identifies the number of bands (steps) used to display a fountain fill. When you create a fountain fill, the space required to blend the colors is divided by the number of fountain steps displayed in the Steps box. This value is optional and the default value is 0.
EdgePad	The **EdgePad** parameter defines the length of solid colors at the beginning and end of the fountain transparency before the start blending with the next color in the fountain fill.You can change the edge pad of linear, radial, and square fountain fills. Conical fountain fills progress in rays, so you can't change their edge pad. Higher values let the colors remain solid longer before blending, causing the colors to spread more quickly. Lower values result in a smooth transformation between the two colors. This value is optional and the default value is 0.
MidPoint	The **MidPoint** is an imaginary line between two colors in a fountain fill. The value of the mid-point represents its position in relation to two fountain fill colors. By changing this value, you can set the point at which two colors in a fountain fill converge. For example, in a two-color fountain fill using the colors black and white, a value of 50 positions the mid-point in the center of the fill so that half of the fill is black and half is white. Increasing the mid-point value to 99 results in a fountain fill dominated by black. Decreasing the mid-point value to 1 results in a fountain fill dominated by white. This value is optional and the default value is 50.

Example

The following code example selects an object in the active document and determines if the object is a curve object. If the selection is a curve, it changes its **Closed** status to True, joining all the segments in the shape to make a self-contained object. A color fountain

transparency is applied to the new closed curve object and using the Fountain property, the angle in the fountain transparency effect is set to 45 degrees:

```
Sub CurveClosed()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
If s.Type = cdrCurveShape Then
    s.Curve.Closed = True
End If
s.Transparency.ApplyFountainTransparency CreateColorEx(5005, 255, 0, 0), _
CreateColorEx(5005, 0, 0, 0), 1
s.Transparency.Fountain.Angle = 45
End Sub
```

{button ,AL(`CLS_Transparency;FNC_ApplyFountainTransparency')} **Related Topics**

Transparency.ApplyPatternTransparency

Function **ApplyPatternTransparency**(ByVal **Type** AS Long, ByVal FileName AS String, ByVal PatternCanvasIndex AS Long, ByRef FrontColor AS **Color**, ByRef EndColor AS **Color**, ByVal TransformWithShape AS Boolean) AS **PatternFill**

Transparency

Description

The **ApplyPatternTransparency** method applies a pattern fill transparency to an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Pattern transparency is a pre-generated, symmetrical image that is repeated over and over, making it extremely useful for creating tiles. You can fill an object completely with one image, but you would more often use a series of repeated images to form a tiled fill. The effect is similar to applying wallpaper to a wall.

In order to use the **ApplyPatternTransparency** method, you must pass several values as parameters:

Parameters	Description
Type	The **Type** parameter identifies the type of pattern fill and returns a value of cdrPatternFillType.
FileName	The **FileName** parameter identifies the full path name of the graphic that is imported into CorelDRAW to be used as a pattern fill. A file name contains the computer's path and the name of the graphic file. This value is optional.
PatternCanvasIndex	The **PatternCanvasIndex** parameter identifies the pattern canvas in a pattern fill effect. A canvas is a blank area that acts as a background for a pattern effect in CorelDRAW. Each canvas has an index number that uniquely identifies the canvas. This value is optional.
FrontColor	The **FrontColor** parameter identifies the foreground color in a pattern fill. This value is optional and the default value is Nothing(0).
EndColor	The **EndColor** parameter identifies the background color in a pattern fill. This value is optional and the default value is Nothing(0).
TransformWithShape	The **TransformWithShape** parameter is a True or False value, indicating if the pattern fill changes to fit its shape when the shape object is altered. If the value is True, the pattern fill will change according to changes in its shape object. This value is optional and the default value is False.

Example

The following code example applies a two color "checkers" pattern fill to the active shape in CorelDRAW. The fill will transform its shape when the object's shape is altered. The path file of the pattern is displayed in a message box, and the frozen status of the transparency displays as well::

```
Sub FillPattern()
Dim ColorFront as Color
Dim ColorBack as Color
Set ColorFront = ActivePalette.Colors(3)
Set ColorBack = ActivePalette.Colors(4)
With ActiveShape.Transparency
.ApplyPatternTransparency cdrTwoColorPattern, "checkers", 1, ColorFront, ColorBack, True
    MsgBox .Pattern.FilePath
    MsgBox .Frozen
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_ApplyPatternTransparency')} **Related Topics**

Transparency.ApplyTextureTransparency

Function **ApplyTextureTransparency**(ByVal **TextureName** AS String, ByVal LibraryName AS String) AS **TextureFill**

Transparency

Description

The **ApplyTextureTransparency** method applies a texture fill transparency to an object in CorelDRAW. Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

A Texture transparency is a random, fractally generated transparency that you can use to give your objects a natural appearance. Transparency handles let you control the block of fractal texture that controls the transparency of the object.

In order to use the **ApplyTextureTransparency** method, you must pass several values as parameters:

Parameters	Description
TextureName	The **TextureName** parameter returns a string value associated with the name of a texture fill in CorelDRAW. For example, "Air Brush" is texture fill in the "Samples" library of CorelDRAW.
LibraryName	The **LibraryName** parameter returns a string value associated with the name of a texture fill pattern library in CorelDRAW. A texture library is a collection of texture fill pattern files in CorelDRAW. There are several texture libraries included in CorelDRAW. For example, "Samples" is a texture library. The **LibraryName** parameter is optional.

Example

The following code example unfreezes the transparency in the active shape and applies a texture transparency. The texture pattern is called "Air Brush" and it is found in the "Samples" texture library. The **Texture** property is set up to transform the texture transparency as the shape is altered, and the transparency is frozen to prevent changes being made to the active shape.

```
Sub TransTexture()
With ActiveShape.Transparency
    .Unfreeze
    .ApplyTextureTransparency "Air Brush", "Samples"
    .Texture.TransformWithShape = True
    .Freeze
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_ApplyTextureTransparency')} **Related Topics**

Transparency.Freeze

Sub **Freeze**()

Transparency

Description

The **Freeze** method freezes a transparency object in CorelDRAW. You can freeze the contents of a transparency to create a bitmap. Once frozen, the contents of the object no longer interact with other objects on the screen. For example, when you move the frozen transparency, its contents don't change.

Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

Example

The following code example unfreezes the transparency in the active shape and applies a texture transparency. The texture pattern is called "Air Brush" and it is found in the "Samples" texture library. The **Texture** property is set up to transform the texture transparency as the shape is altered, and the transparency is frozen to prevent changes being made to the active shape.

```
Sub TransTexture()
With ActiveShape.Transparency
    .Unfreeze
    .ApplyTextureTransparency "Air Brush", "Samples"
    .Texture.TransformWithShape = True
    .Freeze
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Freeze')} **Related Topics**

Transparency.Unfreeze

Sub **Unfreeze**()

Transparency

Description

The **UnFreeze** method removes the freeze of a transparency object in CorelDRAW. You can freeze the contents of a transparency to create a bitmap. Once frozen, the contents of the object no longer interact with other objects on the screen. For example, when you move the frozen transparency, its contents don't change.

Transparency refers to the ability to see through an item. The opposite of transparent is opaque. Setting lower levels of transparency causes higher levels of opacity and less visibility of the underlying items or image.

Example

The following code example unfreezes the transparency in the active shape and applies a texture transparency. The texture pattern is called "Air Brush" and it is found in the "Samples" texture library. The **Texture** property is set up to transform the texture transparency as the shape is altered, and the transparency is frozen to prevent changes being made to the active shape.

```
Sub TransTexture()
With ActiveShape.Transparency
    .Unfreeze
    .ApplyTextureTransparency "Air Brush", "Samples"
    .Texture.TransformWithShape = True
    .Freeze
End With
End Sub
```

{button ,AL(`CLS_Transparency;FNC_Unfreeze')} **Related Topics**

CloneLink properties

CloneLink Legend

- Application
  - BitmapColorMaskLinked
- CloneParent
  - FillLinked
  - OutlineLinked
- Parent
  - ShapeLinked
  - TransformLinked

CloneLink

Class **CloneLink**

Properties Referenced by

The CloneLink class defines the characteristics of CloneLink objects and describes the look and behavior of the objects through its properties and methods. A clone is a copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

You can also clone a special effect that is applied to an object and apply it to other objects. Objects with a cloned effect take on all changes that are made to that effect in the master.

The CloneLink class represents the properties of the actual link between clone and parent objects. Objects are linked in Object Linking and Embedding (OLE) when information from one file (the source file) is inserted in another file (the destination file). The source file is then linked to the destination file. Changes made to the information in the source file can be automatically or manually updated in the destination file.

{button ,AL(`CLS_CloneLink')} **Related Topics**

A copy of an object or an area of an image that is linked to the original object. Most changes made to the original object (the master) are automatically applied to its clones.

You can also clone a special effect that is applied to an object and apply it to other objects. Objects with a cloned effect take on all changes that are made to that effect in the master.

An object that has been copied using the Clone command. Most changes you make to the Master object are automatically applied to the clone.

CloneLink.Application

Property **Application** AS **Application**

CloneLink

Description

The Application property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The Application property returns a Read-Only value.

Example

The following code example displays the current version of the CorelDRAW in a message box:

```
Sub CloneApp()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
With s.CloneLink.Application
    MsgBox "You are using CorelDRAW " & .Version
End With
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_Application')} **Related Topics**

CloneLink.Parent

Property **Parent** AS **Shape**

CloneLink

Description

The Parent property returns a value associated with the properties, methods and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The Parent property returns a Read-Only value.

Example

The following code example displays the name of the CloneLink's parent layer in a message box:

```
Sub ClonelinkParent()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
With s.CloneLink
    MsgBox "The layer of the CloneLink's parent object is " _
    & vbCrLf & vbCrLf & .Parent.Layer.Name
End With
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_Parent')} **Related Topics**

CloneLink.CloneParent

Property **CloneParent** AS **Shape**

CloneLink

Description

The **CloneParent** object returns a value associated with the master object in the clone-master relationship of objects. In order for this property to be effective, there must be a clone-master relationship between objects in the active document and the clone object must be selected object in CorelDRAW. This property specifies characteristics of the selected clone's master object. Specifies attributes of the CloneParent object

The **CloneParent** property returns a Read-Only value.

Example

The following code example selects a clone object in the active document and displays the name of its parent object in a message box:

```
Sub ParentClone()
Dim s As Shape
Set s = ActiveSelection.Shapes(1)
'note that the selected object must be the clone object
MsgBox "The name of the clone's parent object is " & s.CloneLink.CloneParent.Name
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_CloneParent')} **Related Topics**

CloneLink.FillLinked

Property **FillLinked** AS Boolean

CloneLink

Description

The FillLinked property returns a True or False value indicating if the fill attributes of a clone object are linked to its master object. If the FillLinked property is True, the fill attributes of the clone object are linked to the master object. If linked, the clone has the same fill attributes as its master object.

Example

The following code example looks at the shapes in the current selection that are clone objects, and restores several links (Fill, Outline, Shape and Color) between the clone objects and their master objects:

```
Sub FillLink()
Dim s As Shape
For Each s In ActiveSelection.Shapes(1).Clones
'looks at all clone objects in the current selection
    s.CloneLink.FillLinked = True
'links the fill property of clone and master objects in the selection
    s.CloneLink.BitmapColorMaskLinked = True
    s.CloneLink.OutlineLinked = True
    s.CloneLink.ShapeLinked = True
Next s
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_FillLinked')} **Related Topics**

CloneLink.OutlineLinked

Property **OutlineLinked** AS Boolean

CloneLink

Description

The OutlineLinked property returns a True or False value indicating if the outline attributes of a clone object are linked to its master object. If the OutlineLinked property is True, the outline attributes of the clone object are linked to the master object. If linked, the clone has the same outline attributes as its master object.

Example

The following code example looks at the shapes in the current selection that are clone objects, and restores several links (Fill, Outline, Shape and Color) between the clone objects and their master objects:

```
Sub OutlineLink()
Dim s As Shape
For Each s In ActiveSelection.Shapes(1).Clones
'looks at all clone objects in the current selection
    s.CloneLink.OutlineLinked = True
'links the outline property of clone and master objects in the selection
    s.CloneLink.BitmapColorMaskLinked = True
    s.CloneLink.FillLinked = True
    s.CloneLink.ShapeLinked = True
Next s
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_OutlineLinked')} **Related Topics**

CloneLink.ShapeLinked

Property **ShapeLinked** AS Boolean

CloneLink

Description

The ShapeLinked property returns a True or False value indicating if the shape attributes of a clone object are linked to its master object. If the ShapeLinked property is True, the shape attributes of the clone object are linked to the master object. If linked, the clone object has the same shape attributes as its master object.

Example

The following code example looks at the shapes in the current selection that are clone objects, and restores several links (Fill, Outline, Shape and Color) between the clone objects and their master objects:

```
Sub ShapeLink()
Dim s As Shape
For Each s In ActiveSelection.Shapes(1).Clones
'looks at all clone objects in the current selection
    s.CloneLink.ShapeLinked = True
'links the shape property of clone and master objects in the selection
    s.CloneLink.BitmapColorMaskLinked = True
    s.CloneLink.FillLinked = True
    s.CloneLink.OultineLinked = True
Next s
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_ShapeLinked')} **Related Topics**

CloneLink.TransformLinked

Property **TransformLinked** AS Boolean

CloneLink

Description

The TransformLinked property returns a True or False value indicating if the size and shape attributes of a clone object are linked to its master object. If the TransformLinked property is True, the size and shape attributes of the clone object are linked to the master object. If linked, the clone has the same shape and size attributes as its master object.

Example

The following code example looks at the shapes in the current selection that are clone objects, and restores several links (Fill, Outline, Transform, and Color) between the clone objects and their master objects:

```
Sub TransformLink()
Dim s As Shape
For Each s In ActiveSelection.Shapes(1).Clones
'looks at all clone objects in the current selection
    s.CloneLink.TransformLinked = True
'links the size and shape property of clone and master objects in the selection
    s.CloneLink.BitmapColorMaskLinked = True
    s.CloneLink.FillLinked = True
    s.CloneLink.OultineLinked = True
Next s
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_TransformLinked')} **Related Topics**

CloneLink.BitmapColorMaskLinked

Property **BitmapColorMaskLinked** AS Boolean

CloneLink

Description

The BitmapColorMaskLinked property returns a True or False value indicating if the color attributes of a clone object are linked to its master object. The Bitmap Color Mask Docker lets you mask as many as 10 colors in a bitmap. When you mask colors, you can edit them without altering the other colors in an image. You can also save bitmap color masks for future use

If the BitmapColorMaskLinked property is True, the color attributes of the clone object are linked to the master object. If linked, the clone has the same color attributes as its master object.

Example

The following code example looks at the current selection of shapes in the active document. If the BitmapColorMaskLinked property of the cloned selection is broken (i.e. False), a dialog box prompts you to restore the bitmap color mask link. If you select "Yes" in the dialog box, the link is restored and a message displays in a message box:

```
Sub BitmapLink()
Dim s As Shape
Dim ret As VbMsgBoxResult
Set s = ActiveSelection.Shapes(1)
If Not s.CloneLink.BitmapColorMaskLinked Then
'if the link is currently broken
ret = MsgBox("The Bitmap color mask link is broken. Do you want to restore it?", vbYesNo)
'dialog box prompts user for input
    If ret = vbYes Then
    s.CloneLink.BitmapColorMaskLinked = True
    MsgBox "The link is now active."
    End If
End If
End Sub
```

{button ,AL(`CLS_CloneLink;FNC_BitmapColorMaskLinked')} **Related Topics**

AddIns properties

AddIns Legend

- Application
- Collection
- Parent

AddIns methods

AddIns Legend

Attach

AddIns

Class **AddIns**

Properties Methods Referenced by

{button ,AL(`CLS_AddIns')} **Related Topics**

AddIns.Application

Property **Application** AS **Application**

AddIns

Member of **AddIns**

Read-Only

Return value

Returns Application

Example

Example of usage goes here

{button ,AL(`CLS_AddIns;FNC_Application')} **Related Topics**

AddIns.Parent

Property **Parent** AS **Application**

AddIns

Member of **AddIns**

Read-Only

Return value

Returns Application

Example

Example of usage goes here

{button ,AL(`CLS_AddIns;FNC_Parent')} **Related Topics**

AddIns.Collection

Property **Collection** AS Object

AddIns

Member of **AddIns**

Read-Only

Return value

Returns Object

Example

Example of usage goes here

{button ,AL(`CLS_AddIns;FNC_Collection')} **Related Topics**

AddIns.Attach

Function **Attach**(ByVal **Filter** AS **cdrAddinFilter**, ByVal ExecuteCommandPrompt AS String) AS **AddinHook**

AddIns

Member of **AddIns**

Parameters	Description
Filter	Description of 'Filter' goes here (**in**)
ExecuteCommandPrompt	Description of 'ExecuteCommandPrompt' goes here (**in**) Optional

Return value

Returns AddinHook

Example

Example of usage goes here

{button ,AL(`CLS_AddIns;FNC_Attach')} **Related Topics**

AddinHook properties

AddinHook Legend

- Application
  - Filter
  - Index
- Parent

AddinHook events

AddinHook

Execute
New
ShapeCreated
WhileDrawing

AddinHook

Class **AddinHook**

Properties Events Referenced by

{button ,AL(`CLS_AddinHook')} **Related Topics**

AddinHook.Application

Property **Application** AS **Application**

AddinHook

Member of **AddinHook**

Read-Only

Return value

Returns Application

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_Application')} **Related Topics**

AddinHook.Parent

Property **Parent** AS **Application**

AddinHook

Member of **AddinHook**

Read-Only

Return value

Returns Application

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_Parent')} **Related Topics**

AddinHook.Filter

Property **Filter** AS **cdrAddinFilter**

AddinHook

Member of **AddinHook**

Return value

Returns cdrAddinFilter

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_Filter')} **Related Topics**

AddinHook.Index

Property **Index** AS Long

AddinHook

Member of **AddinHook**

Return value

Returns Long

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_Index')} **Related Topics**

AddinHook.Execute

Event **Execute**()

AddinHook

Member of **AddinHook**

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_Execute')} **Related Topics**

AddinHook.New

Event **New**(ByRef **NewDocument** AS **Document**)

AddinHook

Member of **AddinHook**

Parameters	Description
NewDocument	Description of 'NewDocument' goes here (**in**)

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_New')} **Related Topics**

AddinHook.ShapeCreated

Event **ShapeCreated**(ByRef **NewShape** AS **Shape**)

AddinHook

Member of **AddinHook**

Parameters	Description
NewShape	Description of 'NewShape' goes here (**in**)

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_ShapeCreated')} **Related Topics**

AddinHook.WhileDrawing

Event **WhileDrawing**(ByRef **TheShape** AS **Shape**, ByRef **X** AS Long, ByRef **Y** AS Long)

AddinHook

Member of **AddinHook**

Parameters	Description
TheShape	Description of 'TheShape' goes here (**in**)
X	Description of 'X' goes here (**in**,**out**)
Y	Description of 'Y' goes here (**in**,**out**)

Example

Example of usage goes here

{button ,AL(`CLS_AddinHook;FNC_WhileDrawing')} **Related Topics**

Application properties

Application Legend

- ActiveDocument
- ActiveLayer
- ActivePage
- ActivePalette
- ActiveSelection
- ActiveSelectionRange
- ActiveShape
  - ActiveTool
- ActiveWindow
- ActiveWorkspace
- AddIns
- Application
- AppWindow
- ArrowHeads
- Clipboard
- CommandBars
- ConfigPath
- Documents
  - EventsEnabled
- FontList
  - Optimization
- GMSManager
- MainMenu
- OutlineStyles
- Palettes
  - PanoseMatching
- Parent
- Path
- PatternCanvases
- Printers
- PrintJob
- RecentFiles
- SetupPath
- StatusBar
- VBE

- Version
- VersionBuild
- VersionMajor
- VersionMinor
- Visible
- Windows
- Workspaces

Application methods

Application Legend

- cdrMixedDouble
- cdrMixedLong
- cdrMixedSingle
- CorelScript
- CorelScriptTools
- CreateBWColor
- CreateCMYColor
- CreateCMYKColor
- CreateColor
- CreateColorEx
- CreateDocument
- CreateDocumentFromTemplate
- CreateFixedColor
- CreateGrayColor
- CreateHLSColor
- CreateHSBColor
- CreateLabColor
- CreateRegistrationColor
- CreateRGBColor
- CreateShapePoint
- CreateYIQColor
- OpenCorelScriptFile
- ImportWorkspace
- OpenDocument
- Quit

Application

Class Application

Properties Methods Referenced by

The **Application** class defines the characteristics of the main CorelDRAW application and describes the look and behavior of the application through its properties and methods.

The **Application** object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. With the Application object, **CorelScript** and **Application.CorelScript** can be used interchangeably because they represent the same **Application** object.

The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations. CorelDRAW can import versions 5 to 8. The CDR file format is a native of CorelDRAW. It can contain both vectors and bitmaps.

{button ,AL(`CLS_Application')} **Related Topics**

Application.CommandBars

Property **CommandBars** As **CommandBars**

Gets a collection of all command bars

Member of **Application**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Application;FNC_CommandBars')} **Related Topics**

Application.GMSManager

Property **GMSManager** As **GMSManager**

Gets a GMS Manager object

Member of **Application**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Application;FNC_GMSManager')} **Related Topics**

Application.MainMenu

Property **MainMenu** As **CommandBar**

Gets the main menu

Member of **Application**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Application;FNC_MainMenu')} **Related Topics**

Application.StatusBar

Property **StatusBar** As **CommandBar**

Gets the status bar

Member of **Application**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Application;FNC_StatusBar')} **Related Topics**

Application.ImportWorkspace

Sub **ImportWorkspace**(ByVal **FileName** As String)

Imports workspace elements into the current workspace

Member of **Application**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Application;FNC_ImportWorkspace')} **Related Topics**

Application.CreateBWColor

Function **CreateBWColor**(ByVal **White** As Boolean) As **Color**

Description

The **CreateBWColor** method creates a new color based on the Black and White color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Black and White color model is a 1-bit color mode that stores images as two solid colors - black and white - with no gradations. This color mode is useful for line art and simple graphics.

Parameters	Description
White	Sets the appearance of white in the Black and White color model. This value is a True or False value.

Example

```
CreateBWColor (True)
```

{button ,AL(`CLS_Application;FNC_CreateBWColor')} **Related Topics**

Application.CreateCMYColor

Function **CreateCMYColor**(ByVal **Cyan** As Long, ByVal **Magenta** As Long, ByVal **Yellow** As Long) As **Color**

Description

The **CreateCMYColor** method creates a new color based on the CMY color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMY is a color mode made up of cyan (C), magenta (M), and yellow (Y). This mode is used in the three-color printing process. In Corel applications, the CMY mode is the inverse of the RGB mode, with values ranging from 0 to 255. The CMY color mode is based on the CMY color model.

Parameters	Description
Cyan	Sets the level of cyan in the CMY color model. Values range from 0 to 255.
Magenta	Sets the level of magenta in the CMY color model. Values range from 0 to 255.
Yellow	Sets the level of magenta in the CMY color model. Values range from 0 to 255.

Example

```
CreateCMYColor (255, 100, 100)
```

{button ,AL(`CLS_Application;FNC_CreateCMYColor')} **Related Topics**

Application.CreateCMYKColor

Function **CreateCMYKColor**(ByVal **Cyan** As Long, ByVal **Magenta** As Long, ByVal **Yellow** As Long, ByVal **Black** As Long) As **Color**

Description

The **CreateCMYKColor** method creates a new color based on the CMYK color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

CMYK is a color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

Parameters	Description
Cyan	Sets the level of cyan in the CMY color model. Values range from 0 to 255.
Magenta	Sets the level of magenta in the CMY color model. Values range from 0 to 255.
Yellow	Sets the level of magenta in the CMY color model. Values range from 0 to 255.
Black	Sets the level of black in the CMYK color model. Values are percentages and range from 0 to 100.

Example

```
CreateCMYKColor (100, 0, 100, 0)
```

{button ,AL(`CLS_Application;FNC_CreateCMYKColor')} **Related Topics**

Application.CreateDocumentFromTemplate

Function **CreateDocumentFromTemplate**(ByVal **Template** As String, [ByVal IncludeGraphics As Boolean = True]) As **Document**

Description

The **CreateDocumentFromTemplate** method opens a new document from a template in CorelDRAW. A document page is often referred to as a "page". You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

A template is a collection of styles that work together to govern the appearance of a drawing. Each CorelDRAW drawing is based on a template. You can choose a preset template or create your own template.

Parameters	Description
Template	The Template parameter is a string value that identifies a template.
IncludeGraphics	The IncludeGraphics parameter is a True or False value that indicates whether or not to include graphics in the new document. This value is optional and the default value is True.

{button ,AL(`CLS_Application;FNC_CreateDocumentFromTemplate')} **Related Topics**

Application.CreateFixedColor

Function **CreateFixedColor**(ByVal **PaletteID** As **cdrPaletteID**, ByVal **PaletteIndex** As Long, [ByVal Tint As Long = 100]) As **Color**

Description

The **CreateFixedColor** method creates a new color from a fixed color palette in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

Parameters	Description
PaletteID	Sets the Palette ID that uniquely identifies which color palette to use in CorelDRAW.
PaletteIndex	Sets the color to use by referencing the index number of a color within a palette.)
Tint	Sets the tint of the palette color. Tints are lighter shades of a spot color that are created by changing the percentage tint value. This value is optional and the default value is 100.

Example

```
CreateFixedColor ( cdrPANTONECoated, 1, 50)
```

{button ,AL(`CLS_Application;FNC_CreateFixedColor')} **Related Topics**

Application.CreateGrayColor

Function **CreateGrayColor**(ByVal **GrayValue** As Long) As **Color**

Description

The **CreateGrayColor** method creates a new color from the Grayscale color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The Grayscale color model is a color mode that displays images using 256 shades of gray. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white). In the RGB color mode, a grayscale value corresponds to equal amounts of all RGB colors; in CMYK, a grayscale value corresponds to zero C, M, and Y values, with a positive K value; in HSB, a grayscale value corresponds to zero H and S values, with a positive B value. The Grayscale color mode is based on the Grayscale color model.

Parameters	Description
GrayValue	Sets the level of gray in the Grayscale color model. Each color is defined as a value between 0 and 255, where 0 is darkest (black) and 255 is lightest (white).

Example

```
CreateGrayColor (255)
```

{button ,AL(`CLS_Application;FNC_CreateGrayColor')} **Related Topics**

Application.CreateHLSColor

Function **CreateHLSColor**(ByVal **Hue** As Long, ByVal **Lightness** As Long, ByVal **Saturation** As Long) As **Color**

Description

The **CreateHLSColor** method creates a new color from the HLS color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The HLS model is a variation of the HSB model and contains three components: hue, lightness, and saturation. Hue determines color (yellow, orange, red, etc.); lightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). The circular visual selector defines the H value (0 to 360) and the S value (0 to 100); the vertical visual selector defines the L value (0 to 100).

Parameters	Description
Hue	Sets the Hue for the HLS color model. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.
Lightness	Sets the Lightness for the HLS color model. Lightness determines the intensity of the color. Values range from 0 to 100.
Saturation	Sets the Saturation for the HLS color model. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.

Example

```
CreateHLSColor (200, 100, 100)
```

{button ,AL(`CLS_Application;FNC_CreateHLSColor')} **Related Topics**

Application.CreateHSBColor

Function **CreateHSBColor**(ByVal **Hue** As Long, ByVal **Saturation** As Long, ByVal **Brightness** As Long) As **Color**

Description

The **CreateHSBColor** method creates a new color from the HSB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

HSB is a color model that approximates the way the human eye perceives color. In the HSB model, color is defined by three components: hue, saturation, and brightness. Hue determines color (yellow, orange, red, etc.); brightness determines perceived intensity (lighter or darker color); and saturation determines color depth (from dull to intense). In the HSB color model, Hue (H) is expressed as a degree of rotation on a circular color wheel. Saturation (S) and brightness (B) are expressed as percentages of full intensity.

Parameters	Description
Hue	Sets the Hue for the HSB color model. Hue is the property of a color that allows us to classify it by its name. For example, blue, green, and red are all hues.
Saturation	Sets the Saturation for the HSB color model. Saturation is the purity or vividness of a color, expressed as the absence of white. A color that has 100% saturation contains no white. A color with 0% saturation is a shade of gray.
Brightness	Sets the Brightness for the HSB color model. Brightness is the amount of light that is transmitted or reflected from a given pixel. In the HSB color model, brightness is a measure of how much white a color contains. In this case, a brightness value of 0 produces black and a brightness value of 255 produces white.

Example

```
CreateHSBColor (200, 100, 100)
```

{button ,AL(`CLS_Application;FNC_CreateHSBColor')} **Related Topics**

Application.CreateLabColor

Function **CreateLabColor**(ByVal **L** As Long, ByVal **A** As Long, ByVal **B** As Long) As **Color**

Description

The **CreateLabColor** method creates a new color from the Lab color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The LAB color model is a color model created by the Commission Internationale de l'Eclairage (CIE). It contains a luminance (or lightness) component (L) and two chromatic components: "a" (green to red) and "b" (blue to yellow). The Lab color mode is based on the Lab color model.

Parameters	Description
L	Sets the Luminance, or brightness, value in the LAB color model. Values range from 0 to 255.
A	Sets one of two chromatic components in the LAB color model. "A" refers to a range of colors between green and red.
B	Sets one of two chromatic components in the LAB color model. "B" refers to a range of colors between blue and yellow.

Example

```
CreateLabColor (128, 0, 0)
```

{button ,AL(`CLS_Application;FNC_CreateLabColor')} **Related Topics**

Application.CreateRegistrationColor

Function **CreateRegistrationColor**() As **Color**

Description

The **CreateRegistrationColor** method creates a new color based on the Registration color model in CorelDRAW. Color trapping is necessary to compensate for poor color registration that occurs when the printing plates used to print each color, called color separations, are not aligned perfectly. Poor registration causes unintentional white slivers to appear between adjoining colors. Trapping is accomplished by intentionally overlapping colors so that minor problems with alignment are not noticed.

Example

```
CreateRegistationColor
```

{button ,AL(`CLS_Application;FNC_CreateRegistrationColor')} **Related Topics**

Application.CreateRGBColor

Function **CreateRGBColor**(ByVal **Red** As Long, ByVal **Green** As Long, ByVal **Blue** As Long) As **Color**

Description

The **CreateRGBColor** method creates a new color from the RGB color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

RGB is a color mode that contains three components: red (R), green (G), and blue (B). The RGB color mode is based on the RGB color model. In the RGB color mode, a value between 0 and 255 is assigned to each channel of red, green, and blue. An RGB color with the component values 0:25:118, for example, contains no red, some green, and more blue, resulting in a slightly greenish blue color. Monitors, scanners, and the human eye use RGB to produce or detect color.

Parameters	Description
Red	Sets the level of red in the RGB color model. Values range from 0 to 255.
Green	Sets the level of green in the RGB color model. Values range from 0 to 255.
Blue	Sets the level of blue in the RGB color model. Values range from 0 to 255.

Example

```
CreateRGBColor (255, 0, 0)
```

{button ,AL(`CLS_Application;FNC_CreateRGBColor')} **Related Topics**

Application.CreateShapePoint

Function **CreateShapePoint**(ByVal **PositionX** As Double, ByVal **PositionY** As Double) As **ShapePoint**

Description

The **CreateShapePoint** method creates a free point in CorelDRAW. A shape point is a coordinate point in CorelDRAW. A shape point is defined by its X and Y positions, which are established by the CorelDRAW rulers and the document unit of measurement.

A shape point can be part of a shape object in a document, or it can be a free point within the document. A point is free if it stands alone as a single point and is not part of another object in CorelDRAW.

Parameters	Description
PositionX	The **PositionX** property sets the horizontal position of a shape point in CorelDRAW.
PositionY	The **PositionY** property sets the vertical position of a shape point in CorelDRAW.

Example

```
CreateShapePoint (3, 3)
```

{button ,AL(`CLS_Application;FNC_CreateShapePoint')} **Related Topics**

Application.CreateYIQColor

Function **CreateYIQColor**(ByVal **Y** As Long, ByVal **I** As Long, ByVal **Q** As Long) As **Color**

Description

The **CreateYIQColor** method creates a new color from the YIQ color model in CorelDRAW. A color is an effect applied to an object that alters the object's appearance by the way it reflects light. A color model is a system that defines the number and kind of colors that make up a bitmap image. Black-and-White, Grayscale, RGB, CMYK, and Paletted are examples of popular color modes. A color model is a system used to organize and define colors according to a set of basic properties that may be reproduced.

The YIQ color model is a color model used in television broadcast systems (North American video standard - NTSC). Colors are split into a luminance value (Y) and two chromaticity values (I and Q). On a color monitor, all three components are visible. On a monochrome monitor, only the Y component is visible. The square, two-dimensional visual selector defines the I and Q values, and the vertical visual selector defines the Y value. All values are scaled from 0 to 255.

Parameters	Description
Y	Sets the luminance, or brightness, value in the YIQ color model. Values range from 0 to 255.
I	Sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.
Q	Sets one of two chromaticity values in the YIQ color model. In monitor calibration, chromaticity refers to the chroma (hue adjustment) of your monitor. Values range from 0 to 255.

Example

```
CreateYIQColor (100, 255, 255)
```

{button ,AL(`CLS_Application;FNC_CreateYIQColor')} **Related Topics**

Application.Printers

Property **Printers** As Printer

Description

The **Printers** property returns a value associated with a printer that provides print jobs for documents in CorelDRAW. This property returns a Read-Only value.

{button ,AL(`CLS_Application;FNC_Printers')} **Related Topics**

Application.PrintJob

Property **PrintJob** As PrintJob

Description

The **PrintJob** property returns a value associated with a print job in CorelDRAW. This property returns a Read-Only value.

{button ,AL(`CLS_Application;FNC_PrintJob')} **Related Topics**

Application.Application

Property **Application** AS **Application**

Application

Description

The **Application** property returns a value associated with the main CorelDRAW application. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub ApplicationVersion()
    MsgBox "You are using: CorelDRAW " & Version
End Sub
```

{button ,AL(`CLS_Application;FNC_Application')} **Related Topics**

Application.Parent

Property **Parent** AS **Application**

Application

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the application path in a message box:

```
Sub ParentApplication()
    MsgBox .Parent.Path
End Sub
```

{button ,AL(`CLS_Application;FNC_Parent')} **Related Topics**

Application.Visible

Property **Visible** AS Boolean

Application

Description

The **Visible** property returns or sets a True or False value indicating the visibility status of CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

If the **Visible** property is True, the application is running and visible on the monitor display. This property allows you to show or hide the CorelDRAW application window.

Example

The following code example hides CorelDRAW by setting the **Visible** property to False, and then shows CorelDRAW by setting the **Visible** property to True. Message boxes display messages indicating the visibility of the application:

```
Sub ApplicationVisible()
Visible = False
    MsgBox "CorelDRAW is hidden."
Visible = True
    MsgBox "CorelDRAW is now visible."
End Sub
```

{button ,AL(`CLS_Application;FNC_Visible')} **Related Topics**

Application.Documents

Property **Documents** AS **Documents**

Application

Description

The **Documents** property returns information about the **Documents** collection in CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

The **Documents** collection defines the characteristics of document collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

A Document in CorelDRAW is a single page drawing, or a collection of single-page drawings that make up a multi-page document

The **Documents** property returns a Read-Only value.

Example

The following code example checks for open documents. If a document is open, text is added and if there is no open document, a message displays in a message box:

```
Sub ApplicationDocument()
If Documents.Count > 0 Then
    ActiveLayer.CreateArtisticText 0, 0, "Text"
Else
    MsgBox " There is no document open."
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_Documents')} **Related Topics**

Application.CorelScript

Function **CorelScript**() AS Object

Application

Description

The **CorelScript** method creates a Corel SCRIPT object. With CorelDRAW you can create, save, and run a script using the Corel SCRIPT language. This means that when working in CorelDRAW you can automate operations you use frequently by creating and saving a script. For more advanced script writing and editing, you can use the tools provided with the Corel SCRIPT Editor.

Although most CorelDRAW application commands are one-word equivalents of their corresponding menu commands, you might need more than the command itself to execute an action in these applications. If a command needs more information than is provided by the command name alone, parameters are required. Parameters represent aspects of the feature that you can change or selections you can make.

Corel SCRIPT programming statements and functions are a common set of instructions that can be used with any Corel application that supports scripting. Programming statements and functions are derived from traditional BASIC programming language dialects.

Example

The following code example creates a new script object with the **CorelScript** method. The script object creates a new file and a rectangle is added to that new document:

```
Sub ApplicationCorelScript()
Dim cs As Object
Set cs = CorelScript
cs.FileNew
cs.CreateRectangle 100000, -100000, -100000, 100000, 0, 0, 0, 0
End Sub
```

{button ,AL(`CLS_Application;FNC_CorelScript')} **Related Topics**

Any script you create by saving a recording of your CorelDRAW operations is comprised of Corel SCRIPT application commands. Corel SCRIPT application commands instruct CorelDRAW to perform specific actions. For example, a command might instruct CorelDRAW to open or to close a document. The application commands are easy to understand, since most are one-word equivalents of the corresponding Corel application user interface. For example, the .FileNew command creates a new document.

Corel SCRIPT programming statements and functions send instructions or perform actions that are not part of another Corel application. For example, Corel SCRIPT programming statements can be used to display a custom dialog box, include flow control statements and constructs such as loops, create and manipulate variables, and retrieve information about your computer setup. On their own, Corel SCRIPT programming statements form a powerful programming language. A script containing only Corel SCRIPT programming statements can be executed even if another Corel application is not running.

Application.ActiveDocument

Property **ActiveDocument** AS **Document**

Application

Description

The **ActiveDocument** property returns a value associated with the current document in CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

Although several documents may be open in CorelDRAW at one time, the active document is the document that is currently in use.

It is important to note that you do not need to activate a document before working with it in CorelDRAW. A document is automatically activated when it is referenced in a programming structure. For example:

```
Documents(1).Activate
ActiveDocument.AddPages 1
```

These two lines of code produce the same result as:

```
Documents(1).AddPages 1
```

Both of these examples add a new page to the active document, which is the first document.

The **ActiveDocument** property returns a Read-Only value.

Example

The following code example creates two new documents in CorelDRAW. An ellipse is added to the first document and a rectangle is added to the second document. The first document is made the active document with the Activate method and a new page is added to the first document:

```
Sub DocumentActive()
CreateDocument
'creates the first document
ActiveLayer.CreateEllipse2 3, 2, 1
CreateDocument
'creates the second document
ActiveLayer.CreateRectangle 2, 3, 4, 5
Documents(1).Activate
'first document is the active document
ActiveDocument.AddPages 1
'you do not need to activate a document before working with it
'the last two lines of this procedure can be replaced with:
'Documents(1).AddPages 1
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveDocument')} **Related Topics**

Application.ActivePage

Property **ActivePage** AS **Page**

Application

Description

The **ActivePage** property returns a value associated with the active page a CorelDRAW document. A document page is often referred to as a page. You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

The **ActivePage** property returns a Read-Only value.

Example

The following code example changes the background color of the current page in the open document to red:

```
Sub PageActive()
ActivePage.Color.RGBAssign 255, 0, 0
'turns the background color red
End Sub
```

{button ,AL(`CLS_Application;FNC_ActivePage')} **Related Topics**

Application.ActiveWindow

Property **ActiveWindow** AS **Window**

Application

Description

The **ActiveWindow** property returns a value associated with the active window in CorelDRAW. A window contains a CorelDRAW drawing. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. CorelDRAW may have several windows open at one time, but the active window is the window that is currently in use.

The **ActiveWindow** property returns a Read-Only value.

Example

The following code example refreshes the active window (updates it with the most recent information) in the current document:

```
Sub WindowActive()
ActiveWindow.Refresh
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveWindow')} **Related Topics**

Application.Windows

Property **Windows** AS **Windows**

Application

Description

The **Windows** property returns information about the **Windows** collection in CorelDRAW. The **Windows** collection defines the characteristics of windows collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

A Window contains a CorelDRAW drawing. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print.

The **Windows** property returns a Read-Only value.

Examples

The following code example sets the zoom level of all document windows to 200%:

```
Sub WindowsZoom()
Dim w As Window
For Each w In Windows
'makes the change to all windows in the collection
    w.ActiveView.Zoom = 200
Next w
End Sub
```

The following code example tiles all document windows horizontally:

```
Sub WindowsTile()
Windows.Arrange cdrTileHorizontally
End Sub
```

{button ,AL(`CLS_Application;FNC_Windows')} **Related Topics**

Application.CorelScriptTools

Function **CorelScriptTools**() AS Object

Application

Description

The **CorelScriptTools** method creates a CorelSCRIPT tool object. A script tool is implemented to assist with a particular Corel SCRIPT task. With CorelDRAW you can create, save, and run a script using the Corel SCRIPT language. This means that when working in CorelDRAW you can automate operations you use frequently by creating and saving a script. For more advanced script writing and editing, you can use the tools provided with the Corel SCRIPT Editor.

Example

The following code example creates a SCRIPT tool object called a **File Open** dialog box. If there is something entered in the path location of the dialog box, the tool looks for and opens that file. If is does not exist, a message displays in a message box:

```
Sub ScriptTools()
Dim cst As Object
Dim FilePath As String
Set cst = CorelScriptTools
'creates a SCRIPT tool object
FilePath = cst.GetFileBox()
'opens a File Open dialog box
    If FilePath <> "" Then    'makes sure the path is not an empty string
        If cst.FileAttr(FilePath) = 0 Then
        MsgBox "The file path you selected doesn't exist - please try again"
        End If
    End If
End Sub
```

{button ,AL(`CLS_Application;FNC_CorelScriptTools')} **Related Topics**

Application.ActiveWorkspace

Property **ActiveWorkspace** AS **Workspace**

Application

Description

The **ActiveWorkspace** property returns a value associated with the current workspace in CorelDRAW. A workspace is a container for all the files that make up a project. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. Several workspaces may be available in the application, but the active workspace is the workspace that is currently in use.

You can create multiple workspaces for specific users or specific tasks and then apply them when required. You can also delete workspaces when they are no longer needed.

The **ActiveWorkspace** property returns a Read-Only value.

Example

The following code example displays the name of the active workspace in a message box:

```
Sub WorkspaceActive()
    MsgBox ActiveWorkspace.Name
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveWorkspace')} **Related Topics**

Application.Workspaces

Property **Workspaces** AS **Workspaces**

Application

Description

The **Workspaces** property returns information about the **Workspaces** collection in CorelDRAW. The **Workspaces** collection defines the characteristics of Workspace collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

A workspace is a container for all the files that make up a project. A workspace is an enclosed work environment that is ideally suited to meet specific needs when creating a drawing in CorelDRAW. Several workspaces may be available in the application, but the active workspace is the workspace that is currently in use.

You can create multiple workspaces for specific users or specific tasks and then apply them when required. You can also delete workspaces when they are no longer needed.

The **Workspaces** property returns a Read-Only value.

Example

The following code example displays the available workspaces in a message box, with an asterisk next to the name of the active workspace:

```
Sub WorkspaceCollection()
Dim s As String
Dim wks As Workspace
s = "Available Workspaces: "
For Each wks In Workspaces
    s = s & wks.Name
    If wks.Active Then
    s = s & "  *"
    End If
Next wks
MsgBox s
End Sub
```

{button ,AL(`CLS_Application;FNC_Workspaces')} **Related Topics**

Application.ActivePalette

Property **ActivePalette** AS **Palette**

Application

Description

The **ActivePalette** property returns a value associated with the current color palette in CorelDRAW. A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

You can use standard color collections like the Uniform Color Palette, customizable Color Palettes that you create and arrange, or color-matching systems like the PANTONE MATCHING SYSTEM.

It is important to note that even though CorelDRAW may have several color palettes, the active palette is the palette that is currently in use in CorelDRAW.

The **ActivePalette** property returns a Read-Only value.

Example

The following code example displays the name of the default palette, and the number of colors in that palette, in a message box:

```
Sub PaletteActive()
MsgBox "The default color palette is: " & ActivePalette.Name _
& "It contains " & ActivePalette.ColorCount & " colors."
End Sub
```

{button ,AL(`CLS_Application;FNC_ActivePalette')} **Related Topics**

Application.Palettes

Property **Palettes** AS **Palettes**

Application

Description

The **Palettes** property returns information about the **Palettes** collection in CorelDRAW. The **Palettes** collection defines the characteristics of Palette collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

A Color Palette is a collection of solid colors. In CorelDRAW, you can use the on-screen Color Palette, the Select Color dialog box, or the Color Docker to choose colors for fills, outlines, and paper.

You can use standard color collections like the Uniform Color Palette, customizable Color Palettes that you create and arrange, or color-matching systems like the PANTONE MATCHING SYSTEM.

The **Palettes** property returns a Read-Only value.

Example

The following code example displays the name of all palettes in use, and the number of colors in each palette, in a message box:

```
Sub PalettesList()
Dim pal As Palette
Dim s As String
s = "Active palettes: "
For Each pal In Palettes
    s = s & pal.Name & " (" & pal.ColorCount & " colors)"
Next pal
    MsgBox s
End Sub
```

{button ,AL(`CLS_Application;FNC_Palettes')} **Related Topics**

Application.Quit

Sub **Quit**()

Application

Description

The **Quit** method closes the current session of CorelDRAW. You will be prompted to save any and all changes when you execute the **Quit** command. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

Example

The following code example terminates the current session of CorelDRAW:

```
Sub QuitApp()
Quit
End Sub
```

{button ,AL(`CLS_Application;FNC_Quit')} **Related Topics**

Application.CreateColor

Function **CreateColor**() AS **Color**

Application

Description

The **CreateColor** method creates a color object in CorelDRAW. When this method is used to create a color object, it creates the object with default color properties. You can change the default outline and fill colors by choosing a color when no object is selected. A dialog box prompts you to select the type of object for which you want to change the default color.

Examples

The following code examples show several ways to apply a cyan fill to the first selected object in the active document:

```
Sub CreateColor1()
Dim c As Color
Set c = CreateColor
'this function is obsolete since Color is now created with the NEW keyword
c.CMYKAssign 500, 100, 0, 0
ActiveDocument.Selection.Shapes(1).Fill.ApplyUniformFill c
End Sub

Sub CreateColor2()
Dim c As New Color
c.CMYKAssign 500, 100, 0, 0
ActiveDocument.Selection.Shapes(1).Fill.ApplyUniformFill c
End Sub

Sub CreateColor3()
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.CMYKAssign 500, 100, 0, 0
End Sub
```

{button ,AL(`CLS_Application;FNC_CreateColor')} **Related Topics**

Application.OpenCorelScriptFile

Function **OpenCorelScriptFile**(ByVal **FileName** AS String) AS **CorelScriptFile**

Application

Description

The **OpenCorelScriptFile** method opens an existing Corel SCRIPT file and runs the script from a VBA macro. A macro is a collection of automatic tasks performing an action. It is a symbol, name or key that represents a list of commands.

With the **OpenCorelScriptFile** method, you must specify the script file by passing the file name in a parameter:

Parameters	Description
FileName	The **FileName** parameter identifies the full path name of the Corel SCRIPT file that is opened up in CorelDRAW and executed by converting it to a VBA macro.

Example

The following code example opens an existing SCRIPT file. An ellipse is created on the active layer of the current document and the script runs to create a fill effect on the ellipse:

```
Sub ScriptFileOpen()
Dim csf As CorelScriptFile
Set csf = OpenCorelScriptFile("C:\Corel\Graphics10\Draw\Fill_Out\Fills\Fountain\aliensky.csc")
ActiveLayer.CreateEllipse2 3, 3, 2
csf.Play
End Sub
```

{button ,AL(`CLS_Application;FNC_OpenCorelScriptFile')} **Related Topics**

A macro is a collection of automatic tasks performing an action. It is a symbol, name or key that represents a list of commands.

Application.FontList

Property **FontList** AS **FontList**

Application

Description

The **FontList** property returns information about the **FontList** collection in CorelDRAW. The **FontList** collection defines the characteristics of Font collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

The **FontList** property returns the name of a font, as identified by its index number. Each font in CorelDRAW is uniquely identified by its index number.

For example, `FontList(4)` returns the name of the fourth font type in the CorelDRAW font collection.

The **FontList** property returns a Read-Only value.

Example

The following code example displays the name of the fourth font in the CorelDRAW font collection in a message box:

```
Sub ListFontName()
MsgBox FontList(4)
End Sub
```

{button ,AL(`CLS_Application;FNC_FontList')} **Related Topics**

Application.AppWindow

Property **AppWindow** AS **AppWindow**

Application

Description

The **AppWindow** property returns a value associated with the application window of CorelDRAW. The application window is the area of CorelDRAW that contains the toolbars, workspaces, and windows. It is the container for the CorelDRAW application. Specifies attributes of the main application window in the CorelDRAW 10.0 application

The **AppWindow** property returns a Read-Only value.

Example

The following code example displays the caption of the main application window in a message box:

```
Sub CaptionAppWindow()
MsgBox AppWindow.Caption
End Sub
```

{button ,AL(`CLS_Application;FNC_AppWindow')} **Related Topics**

Application.RecentFiles

Property **RecentFiles** AS **RecentFiles**

Application

Description

The **RecentFiles** property returns a value associated with the recent file list in CorelDRAW. This list is located under File on the menu bar, directly above the Exit command. It provides quick and easy access to any file that was recently used in CorelDRAW.

The **RecentFiles** property returns a Read-Only value.

Example

The following code example displays a list of all recent files, with full file names, in a message box:

```
Sub RecentFilesList()
Dim s As String
Dim rf As RecentFile
s = "Recent Files" & vbCrLf
For Each rf In RecentFiles
    s = s & vbCrLf & rf.Name & " (" & rf.FullName & ")"
Next rf
    MsgBox s
End Sub
```

{button ,AL(`CLS_Application;FNC_RecentFiles')} **Related Topics**

Application.VBE

Property **VBE** AS Object

Application

Description

The **VBE** property returns a reference to the VBA Editor in CorelDRAW. The VBA Editor is the component that creates, stores and edits VBA code modules in CorelDRAW. A VBA code module is a programming package of Visual Basic for Applications code segments.

VBA is a subset of the Microsoft Visual Basic (VB) object-oriented programming environment. VBA uses the Visual Basic Editor interactive development environment and the VB programming language to enhance applications by manipulating the application's objects, exposed by its object model. VBA can access other applications from within CorelDRAW 10.0 by referencing that application's object model components.

The **VBE** property returns a Read-Only value.

Example

The following code example displays the editor type name in the Debug window of the VBA Editor in CorelDRAW:

```
Sub AppVBE()
Dim obj as Object
Set obj = Application.VBE
    Debug.Print TypeName (obj)
End Sub
```

{button ,AL(`CLS_Application;FNC_VBE')} **Related Topics**

Application.cdrMixedDouble

Function **cdrMixedDouble**() AS Double

Application

Description

The **cdrMixedDouble** property returns or sets a "Double" value for object that contain mixed (uneven) settings in CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

For example, text that contains font characters of different sizes has mixed settings.

Double is a data type. A data type is a categorization of the information contained within a programming statement. The Double data type, or double-precision floating point, refers to positive and negative numerical values that range between -1.79769313486232E308 and -4.94065645841247E-3214, as well as 4.94065645841247E-324 and 1.79769313486232E308.

Example

The following code example checks to see if the specified selection is a text object. If this text object contains characters with varying font sizes, it sets a uniform font size for the entire text object:

```
Sub MixedDouble()
Dim s As Shape
Set s = ActiveDocument.Selection.Shapes(1)
    If s.Type = cdrTextShape Then
        If s.Text.FontProperties.Size = cdrMixedDouble Then
        s.Text.FontProperties.Size = 12
        End If
    End If
End Sub
```

{button ,AL(`CLS_Application;FNC_cdrMixedDouble')} **Related Topics**

Application.cdrMixedSingle

Function **cdrMixedSingle**() AS Single

Application

Description

The **cdrMixedSingle** property returns or sets a "Single" value for object that contain mixed (uneven) settings in CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

For example, text that contains font characters of different sizes has mixed settings.

Single is a data type. A data type is a categorization of the information contained within a programming statement. The Single data type, or single-precision floating point, refers to positive and negative numerical values that range between -3.402823E38 and -1.401298E-45, as well as 1.401298E-45 and 3.402823E38.

Example

The following code example checks to see if the specified selection is a text object. If this text object contains characters with varying font sizes, it sets a uniform font size for the entire text object:

```
Sub MixedSingle()
Dim s As Shape
Set s = ActiveDocument.Selection.Shapes(1)
    If s.Type = cdrTextShape Then
        If s.Text.FontProperties.Size = cdrMixedSingle Then
        s.Text.FontProperties.Size = 12
        End If
    End If
End Sub
```

{button ,AL(`CLS_Application;FNC_cdrMixedSingle')} **Related Topics**

Application.cdrMixedLong

Function **cdrMixedLong**() AS Long

Application

Description

The **cdrMixedLong** property returns or sets a "Long" value for object that contain mixed (uneven) settings in CorelDRAW. The application object refers to the CorelDRAW application. CorelDRAW is a vector-based drawing application that lets you create professional artwork, from simple logos to intricate technical illustrations.

For example, text that contains font characters of different sizes has mixed settings.

Long is a data type. A data type is a categorization of the information contained within a programming statement. The Long data type, or long integer, refers to positive and negative numerical values that range between -2,147,483,648 and 2,147,483,647.

Example

The following code example checks to see if the specified selection is a text object. If this text object contains characters with varying font sizes, it sets a uniform font size for the entire text object:

```
Sub MixedLong()
Dim s As Shape
Set s = ActiveDocument.Selection.Shapes(1)
    If s.Type = cdrTextShape Then
        If s.Text.FontProperties.Size = cdrMixedLong Then
        s.Text.FontProperties.Size = 12
        End If
    End If
End Sub
```

{button ,AL(`CLS_Application;FNC_cdrMixedLong')} **Related Topics**

Application.EventsEnabled

Property **EventsEnabled** AS Boolean

Application

Description

The **EventsEnabled** property returns a True or False value indicating the enabled status of an event in CorelDRAW. An event is enabled if it is ready and able to execute in the application. If an event is enabled, its **EventsEnabled** status is true. If the event is disabled, the status is set to false. Events are often disabled to temporarily improve performance and speed within CorelDRAW.

The **EventsEnabled** property returns a Boolean value.

Example

The following code example creates an ellipse in the active layer of the current document. The state (which determines the enabled status of the application) is initially enabled, but is declared False, disabling any events that occur prior to the disabled declaration. A second ellipse is created on the active layer and the state is reset to True, enabling all events:

```
Sub EnabledEvents()
Dim state As Boolean
ActiveLayer.CreateEllipse2 0, 0, 5
state = EventsEnabled
'current state is enabled
EventsEnabled = False
'enabled state is now false, meaning all above events are disabled
ActiveLayer.CreateEllipse2 3, 3, 5
EventsEnabled = state
End Sub
```

{button ,AL(`CLS_Application;FNC_EventsEnabled')} **Related Topics**

CDR - CorelDRAW

PAT = Pattern File

CDT - CorelDRAW Template

CMX - Corel Presentation Exchange 6/7

AI - Adobe Illustrator

PS, PRN, EPSE - Post Script Interpreted

WPG - Corel WordPerfect Graphic

WMF - Windows Metafile

EMF - Enhanced Windows Metafile

CGM - Computer Graphics Metafile

PDF - Adobe Portable Document Format

HTM - Hypertext Markup Language

PCT - Macintosh PICT

DXF - AutoCAD

BWG - AutoCAD Drawing

PLT - HPGL Plotter File

VSD - Visio

CMX - Corel Presentation Exchange 5.0

CPX - Corel CMX Compressed

CDX - CorelDRAW Compressed

Application.OpenDocument

Function **OpenDocument**(ByVal **FileName** AS String) AS **Document**

Application

Description

The **OpenDocument** method opens an existing document in CorelDRAW. This document can be one of many document types and the document is opened into the main application window.

With the **OpenDocument** method, you must specify the document file by passing the file name in a parameter:

Parameters	Description
FileName	The **FileName** parameter identifies the full path name of the existing document that is opened up in CorelDRAW.

Example

The following code example opens an existing CorelDRAW (CDR) document, adds a page to the document, creates a colored ellipse on the new page, and saves the entire document:

```
Sub DocumentOpen()
Dim doc As Document
Set doc = OpenDocument("C:\Flower.cdr")
With doc
    .AddPages (1)
    .ActiveLayer.CreateEllipse(0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
    .Save
End With
End Sub
```

{button ,AL(`CLS_Application;FNC_OpenDocument')} **Related Topics**

Application.CreateDocument

Function **CreateDocument**() AS **Document**

Application

Description

The **CreateDocument** method creates a new CorelDRAW document. Even though CorelDRAW supports many document types, the **CreateDocument** method creates a new CorelDRAW (.cdr) document.

Example

The following code example creates a new CorelDRAW document with the **CreateDocument** method, adds a colored ellipse to the active layer of the document, and saves the new document with a new name:

```
Sub DocumentCreate()
Dim doc As Document
Set doc = CreateDocument()
'creates a new CorelDRAW document
doc.ActiveLayer.CreateEllipse (0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
'creates an ellipse in the active layer of the new document with a uniform fill color
doc.saveas "My New Document"
End Sub
```

{button ,AL(`CLS_Application;FNC_CreateDocument')} **Related Topics**

A color mode made up of cyan (C), magenta (M), yellow (Y), and black (K). In the CMYK color mode, color values are expressed as percentages, so a value of 100 for an ink means that it is applied at full saturation. Used in most full-color commercial printing, CMYK is like CMY, but the addition of black (K) allows for true blacks and a wider tonal range. The CMYK color mode is based on the CMYK color model.

cdrPantone=1

cdrCMYK=2

cdrCMY=4

cdrRGB=5

cdrHSB=6

cdrHLS=7

cdrBlackAndWhite=8

cdrGray=9

cdrColorPantone=1

cdrColorCMYK=2

cdrColorCMY=4

cdrColorRGB=5

cdrColorHSB=6

cdrColorHLS=7

cdrColorBlackAndWhite=8

cdrColorGray=9

cdrColorYIQ=11

cdrColorLab=12

cdrColorPantoneHex=14

cdrColorRegistration=20

cdrColorSpot=25

cdrColorMixed=99

Application.CreateColorEx

Function **CreateColorEx**(ByVal ColorModel AS Long, ByVal V1 AS Long, ByVal V2 AS Long, ByVal V3 AS Long, ByVal V4 AS Long, ByVal V5 AS Long, ByVal V6 AS Long, ByVal V7 AS Long) AS **Color**

Application

Description

The **CreateColorEx** method creates a new color, with specified color information, in CorelDRAW. This color is added to the active palette.

With the **CreateColorEx** method, you must specify the color model and components by passing them in a parameter:

Parameters	Description
ColorModel	The **ColorModel** parameter identifies the color model used to create the new color. This parameter specifies the numeric variable that is assigned to the color model. A color model is a simple color chart that defines the range of colors displayed in a color mode.
V1	The **V1** parameter specifies the numeric variable that is assigned to the first color component of the selected color model. For example, cyan is the first color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V2	The **V2** parameter specifies the numeric variable that is assigned to the second color component of the selected color model. For example, magenta is the second color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V3	The **V3** parameter specifies the numeric variable that is assigned to the third color component of the selected color model. For example, yellow is the third color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V4	The **V4** parameter specifies the numeric variable that is assigned to the fourth color component of the selected color model. For example, black is the fourth color component of the CMYK color model. The value is a percentage and can range from 1 to 100. A color that is selected at 100 is applied with full saturation.
V5	The **V5** parameter specifies the numeric variable that is assigned to the fifth color component of the selected color model. Even though the most popular color models use only four levels of color saturations, CorelDRAW has the ability to create colors using seven color channels.
V6	The **V6** parameter specifies the numeric variable that is assigned to the sixth color component of the selected color model. Even though the most popular color models use only four levels of color saturations, CorelDRAW has the ability to create colors using seven color channels.
V7	The **V7** parameter specifies the numeric variable that is assigned to the seventh color component of the selected color model. Even though the most popular color models use only four levels of color saturations, CorelDRAW has the ability to create colors using seven color channels.

Example

The following code example adds a new color to the active color palette using the **CreateColorEx** method. The new color is added to the active color palette in CorelDRAW:

```
Sub CreatePaletteColor()
ActivePalette.AddColor CreateColorEx(5002, 90, 90, 0, 0)
End Sub
```

{button ,AL(`CLS_Application;FNC_CreateColorEx')} **Related Topics**

A channel is an 8-bit grayscale version of your image that functions like a plate used in the commercial printing process: each channel represents one level of color in your image. When all the channels are printed together, they produce the entire range of colors in the image.
For example, an RGB image comprises three channels (red, green, and blue). When all three channels are printed or displayed together, they create the entire range of colors in the image.

Application.ArrowHeads

Property **ArrowHeads** AS **ArrowHeads**

Application

Description

The **Arrowheads** property returns information about the **Arrowheads** collection in CorelDRAW. The **Arrowheads** collection defines the characteristics of **Arrowhead** collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

An arrowhead is a shape found at the end of a line or curve outline in CorelDRAW. Arrowheads may be traditional arrow shape or a variety of other shapes.

The **Arrowheads** property returns a Read-Only value.

Example

The following code example applies an arrowhead with an index of 5 to the beginning of each selected object:

```
Sub ItemArrowHead()
Dim s As Shape
For Each s In ActiveDocument.Selection.Shapes
    If s.Outline.Type = cdrOutline Then
    s.Outline.StartArrow = ArrowHeads.Item(5)
'****can also be referenced as s.Outline.StartArrow = ArrowHeads (5)
    End If
Next s
End Sub
```

{button ,AL(`CLS_Application;FNC_ArrowHeads')} **Related Topics**

cdrNoOutline=0

cdrOutline=1

Application.OutlineStyles

Property **OutlineStyles** AS **OutlineStyles**

Application

Description

The **OutlineStyles** property returns a value associated with the **OutlineStyles** collection object in CorelDRAW. An outline style is a collection of attributes that are assigned to an object's outline. You can assign custom line styles or choose presets that apply a solid, dashed, dotted, or dashed-and-dotted line style.

CorelDRAW comes with many different line styles. Line styles are preset lines that have different attributes, such as dotted lines, or dashed lines. Applying a line style does not change the shape of the line or the amount of the space it occupies on the drawing page.

The **OutlineStyles** property returns a Read-Only value.

Example

The following code example applies predefined outline styles to selected objects. The style that is applied depends on the type of shape selected - if the shape is a rectangle, the first preset outline style is applied to the outline of the rectangle. Only shapes that have an outline are included the selection:

```
Sub StylesOutline()
Dim s As Shape
For Each s In ActiveSelection.Shapes
    If s.Outline.Type = cdrOutline Then
    Select Case s.Type
    Case cdrRectangleShape
        s.Outline.Style = OutlineStyles(1)   'first preset outline style
    Case cdrEllipseShape
        s.Outline.Style = OutlineStyles(2)   'second preset outline style
    Case cdrPolygonShape
        s.Outline.Style = OutlineStyles(3)   'third preset outline style
    End Select
    End If
Next s
End Sub
```

{button ,AL(`CLS_Application;FNC_OutlineStyles')} **Related Topics**

Application.Version

Property **Version** AS String

Application

Description

The **Version** property returns a value associated with the current version of CorelDRAW. A version is an edition of CorelDRAW - it is a particular form of the application, differing from previous editions. The **Version** property contains the version major number (before the decimal) and version minor number (after the decimal) of version numbers.

The **Version** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub AppVersion()
    MsgBox "You are using: " & Version
End Sub
```

{button ,AL(`CLS_Application;FNC_Version')} **Related Topics**

Application.VersionMajor

Property **VersionMajor** AS Long

Application

Description

The **VersionMajor** property returns a value associated with the current version of CorelDRAW. A version is an edition of CorelDRAW - it is a particular form of the application, differing from previous editions. The **VersionMajor** property returns only the first part of the version number (to the left of the decimal place).

For example, if the current version is 10.396, the major version value would be 10.

The **VersionMajor** property is a Read-Only value.

Example

The following code example checks the current version of CorelDRAW to ensure that the version number is at least 10.200:

```
Sub MajorVersion()
If VersionMajor = 10 And VersionMinor < 200 Then
    MsgBox "This script required that CorelDRAW version 10.200 or later be installed."
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_VersionMajor')} **Related Topics**

Application.VersionMinor

Property **VersionMinor** AS Long

Application

Description

The **VersionMinor** property returns a value associated with the current version of CorelDRAW. A version is an edition of CorelDRAW - it is a particular form of the application, differing from previous editions. The **VersionMinor** property returns only the secondt part of the version number (to the right of the decimal place).

For example, if the current version is 10.396, the minor version value is .396.

The **VersionMinor** property is a Read-Only value.

Example

The following code example checks the current version of CorelDRAW to ensure that the version number is at least 10.200:

```
Sub MinorVersion()
If VersionMajor = 10 And VersionMinor < 200 Then
    MsgBox "This script required that CorelDRAW version 10.200 or later be installed."
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_VersionMinor')} **Related Topics**

Application.VersionBuild

Property **VersionBuild** AS Long

Application

Description

The **VersionBuild** property returns a value associated with the current version of CorelDRAW. A version is an edition of CorelDRAW - it is a particular form of the application, differing from previous editions. The **VersionBuild** property returns only the build number of the current version. A build is specific edition of an application within a version in an application development environment.

The **VersionBuild** property is a Read-Only value.

Examples

The following code example displays the build number of the current CorelDRAW application in a message box:

```
Sub BuildVersion()
MsgBox "You are using CorelDRAW build " & VersionBuild
End Sub
```

Version control was only introduced in CorelDRAW 9 Office Edition. To detect earlier versions, try the following code example:

```
Sub OldVersion()
Dim v&, b&
GetDrawVersion v, b
MsgBox v & "." & b
End Sub
```

```
Sub GetDrawVersion()
Dim v As Long
Dim b As Long
On Error Resume Next
Err.Clear
v = CorelDRAW.Application.VersionMajor
If Err.Number <> 0 Then
    v = 9
    b = 439
Else
    b = CorelDRAW.Application.VersionBuild
End If
On Error GoTo 0
    MsgBox "You are using CorelDRAW " & v & "." & b
End Sub
```

{button ,AL(`CLS_Application;FNC_VersionBuild')} **Related Topics**

Application.Path

Property **Path** AS String

Application

Description

The **Path** property returns a string value associated with the full path identity of the CorelDRAW directory. A path is the physical folder location of the application on a hard drive.

The **Path** property returns a Read-Only value.

Example

The following code example displays the path of the CorelDRAW directory in a message box:

```
Sub PathApplication()
MsgBox Path
End Sub
```

{button ,AL(`CLS_Application;FNC_Path')} **Related Topics**

Application.ConfigPath

Property **ConfigPath** AS String

Application

Description

The **ConfigPath** property returns a string value associated with the location of the configuration directory of CorelDRAW. The configuration directory contains the software settings used to customize an application. .

The **ConfigPath** property returns a Read-Only value.

Example

The following code example displays the path of the CorelDRAW configuration directory in a message box:

```
Sub PathConfig()
    MsgBox ConfigPath
End Sub
```

{button ,AL(`CLS_Application;FNC_ConfigPath')} **Related Topics**

Application.SetupPath

Property **SetupPath** AS String

Application

Description

The **SetupPath** property returns a string value associated with the location of the setup directory of CorelDRAW. The setup directory contains the software settings used to install an application. .

The **SetupPath** property returns a Read-Only value.

Example

The following code example displays the path of the CorelDRAW setup directory in a message box:

```
Sub PathSetup()
MsgBox SetupPath
End Sub
```

{button ,AL(`CLS_Application;FNC_SetupPath')} **Related Topics**

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Once active, a layer is ready to receive any new objects you draw, import, or paste onto it.

Application.ActiveLayer

Property **ActiveLayer** AS **Layer**

Application

Description

The **ActiveLayer** property returns a value associated with the active layer of CorelDRAW. A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

Once active, a layer is ready to receive any new objects you draw, import, or paste onto it.

The **ActiveLayer** property returns a Read-Only value.

Example

The following code example creates an ellipse on the active layer in the current document:

```
Sub LayerActive()
ActiveLayer.CreateEllipse 3, 3, 2, 1
ActiveDocument.Selection.Shapes(1).Fill.UniformColor.CMYKAssign 100, 0, 100, 0
'Fills the ellipse green
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveLayer')} **Related Topics**

Application.ActiveSelection

Property **ActiveSelection** AS **Shape**

Application

Description

The **ActiveSelection** property returns a value associated with the currently selected objects in CorelDRAW. Before you can move or shape an object, you must select it. When you select an object, a selection box appears around the object. A selection box appears as an invisible rectangle having eight black squares, called selection handles, arranged at the corners and midpoints, and a small "x" in the center.

If necessary, you can perform actions on more than one object by selecting all the objects you want. When you select multiple objects, a single selection box encloses all the objects and the "x" appears in the center of the selection box. You can select objects using the mouse, keyboard, or menu commands.

You can select objects on any unlocked layer as long as the Edit Across Layers option is enabled in the Object Manager. If this option is disabled, then you can only select objects on the active layer If you select a locked object, the selection handles appear as padlocks. You cannot edit a locked object. To make changes, you must first unlock the object.

The **ActiveSelection** property returns a Read-Only value.

Example

The following code example colors the selected shapes green, if the shape selected is an ellipse:

```
Sub SelectionActive()
Dim s As Shape
For Each s In ActiveSelection.Shapes  'looks at all selected shapes
    If s.Type = cdrEllipseShape Then  'if the shape is an ellipse
    s.Fill.UniformColor.CMYKAssign 100, 0, 100, 0    'Fills the ellipse green
    End If
Next s
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveSelection')} **Related Topics**

Application.PatternCanvases

Property **PatternCanvases** AS **PatternCanvases**

Application

Description

The **PatternCanvases** property returns a value associated with the **PatternCanvases** collection in CorelDRAW. The **PatternCanvases** collection defines the characteristics of pattern canvas collection objects and describes the look and behavior of the collections objects through its properties, methods, and events.

Pattern fills are preset, symmetrical images that can be tiled. You can import bitmaps or vector graphics for use as pattern fills, or you can create simple two-color bitmap patterns. The effect you create is similar to that created by applying wallpaper to a wall. There are three types of pattern fills: two-color, full-color, and bitmap. A canvas contains a pattern fill. The background is like a canvas on top of which characters and graphics are placed.

The **PatternCanvas** property returns a Read-Only value.

Example

The following code example applies a predefined pattern fill, overlaid with a checkerboard pattern, to an ellipse:

```
Sub CanvasPattern()
Dim s As Shape
Dim pf As PatternFill
Dim cnv As New PatternCanvas
cnv.PutCopy PatternCanvases(2)        'copies the second preset canvas pattern
Set s = ActiveLayer.CreateEllipse(8, 7, 3, 4)
    s.Fill.ApplyNoFill
Set pf = s.Fill.ApplyPatternFill(cdrTwoColorPattern, , , CreateColorEx(5005, 255, 0, 0), _
    CreateColorEx(5005, 255, 255, 0)) 'creates a new pattern fill
    pf.Canvas = cnv    'combines canvas and pattern to create a new fill effect
End Sub
```

{button ,AL(`CLS_Application;FNC_PatternCanvases')} **Related Topics**

Application.Clipboard

Property **Clipboard** AS **Clipboard**

Application

Description

The **Clipboard** property returns a value associated with the system clipboard on the local computer. The clipboard is a temporary storage area that is used to hold cut or copied information. The clipboard is always present in Windows and it stores information until it is replaced by another object or selection that has been cut or copied, or until you quit the current session of Windows.

The **Clipboard** property returns a Read-Only value.

Example

The following code example checks to see if there is any data in the system clipboard. If data is present, it is pasted into the current document. If no data is present, a message displays in a message box:

```
Sub ClipboardData()
If Not Clipboard.Empty Then   'if there is data in the clipboard
    ActiveLayer.Paste  'paste data into the active layer
Else
    MsgBox "There is no data in the clipboard."
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_Clipboard')} **Related Topics**

Application.ActiveSelectionRange

Property **ActiveSelectionRange** AS **ShapeRange**

Application

Description

The **ActiveSelectionRange** property returns a value associated with a **ShapeRange** object in CorelDRAW. A **ShapeRange** is a collection of shape objects. The **ShapeRange** defines the area to include in the active view of CorelDRAW..

The **ActiveSelectionRange** returns a Read-Only value.

Example

The following code example displays messages about selected objects in the active document, depending on the number of objects selected:

```
Sub SelectionRangeActive()
If ActiveSelectionRange.Count = 1 Then
    MsgBox "There is only one shape selected in the shape range."
ElseIf ActiveSelectionRange.Count = 0 Then
    MsgBox "There are no shapes selected in the shape range."
Else
    MsgBox "There are " & ActiveSelectionRange.Count & " shapes in the selected shape range."
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveSelectionRange')} **Related Topics**

cdrToolNone=0
cdrToolPick=1
cdrToolNodeEdit=2
cdrToolKnife=64
cdrToolBezierKnife=81
cdrToolEraser=68
cdrToolDrawRectangle=3
cdrToolDrawEllipse=4
cdrToolDrawText=7
cdrToolDrawFreehand=5
cdrToolDrawNaturalPen=103
cdrToolDrawBezier=6
cdrToolHorizontalDimension=54
cdrToolVerticalDimension=55
cdrToolAutoDimension=110
cdrToolSlantedDimension=56
cdrToolLeaderText=57
cdrToolAngledDimension=58
cdrToolConnectorLines=59
cdrToolDrawPolygon=60
cdrToolDrawSpiral=62
cdrToolDrawGrid=63
cdrToolZoom=66
cdrToolPan=67
cdrToolFill=71
cdrToolTransparency=72
cdrToolInteractiveExtrude=73
cdrToolBlend=74
cdrToolRotate=75
cdrToolReflect=76
cdrToolScale=77
cdrToolSkew=78
cdrToolDistortion=79
cdrToolContour=113
cdrToolInsertHTMLFormObject=111
cdrToolDropShadow=80
cdrToolDrawConnector=115
cdrToolEyeDropper=119

Application.ActiveTool

Property **ActiveTool** AS **cdrTools**

Application

Description

The **ActiveTool** property returns or sets a value associated with the currently used tool in CorelDRAW. A tool is a basic drawing instrument available in CorelDRAW that is used to create shapes, lines, and objects. The active tool is the instrument that is currently selected in CorelDRAW.

The **ActiveTool** property returns a value of cdrTools.

Example

The following code example changes the active tool from the Pick tool to the Eye Dropper tool:

```
Sub ToolActive()
If ActiveTool = cdrToolPick Then
    ActiveTool = cdrToolEyeDropper
End If
End Sub
```

{button ,AL(`CLS_Application;FNC_ActiveTool')} **Related Topics**

The Pick tool lets you select and transform objects.

The Eyedropper tool lets you select a fill from an object on the Drawing Window using the mouse.

Application.ActiveShape

Property **ActiveShape** AS **Shape**

Application

Description

The **ActiveShape** property returns a value associated with the active shape in CorelDRAW. A shape is an object that can be displayed as several variations of a rectangle or a circle. The active shape is the shape object that is currently selected in CorelDRAW.

The **ActiveShape** property returns a Read-Only value.

Example

{button ,AL(`CLS_Application;FNC_ActiveShape')} **Related Topics**

Application.Optimization

Property **Optimization** AS Boolean

Application

Description

The **Optimization** property sets the optimization value of the application to True or False. Optimizing an application process means that you fine-tune a program so that it runs more quickly or takes up less space. If the value is set to true, the shape creation process speeds up when creating multiple shapes.

Setting the **Optimization** property to True disables some internal updating. Width and height properties may show incorrect values if changed while optimization is enabled. It is important to disable optimization when exiting a macro, or CorelDRAW 10.0 may not update its screens.

The **Optimization** property returns a Boolean value.

Example

The following code example creates 100 random circles in the active document, with optimization enabled for increased performance:

```
Sub Optimize()
Dim i As Long
Dim x As Double, y As Double, r As Double
Dim n As Long
Dim num As Long
Dim maxx As Double, maxy As Double, maxr As Double
maxx = ActivePage.SizeWidth
maxy = ActivePage.SizeHeight
maxr = 1
num = ActivePalette.ColorCount
Optimization = True
For i = 1 To 100
    x = Rnd() * maxx
    y = Rnd() * maxy
    r = Rnd() * maxr
    n = CLng(Fix(Rnd() * num)) + 1
    Set s = ActiveLayer.CreateEllipse2(x, y, r)
    s.Fill.ApplyUniformFill ActivePalette.Color(n)
Next i
Optimization = False
ActiveWindow.Refresh
End Sub
```

{button ,AL(`CLS_Application;FNC_Optimization')} **Related Topics**

Application.PanoseMatching

Property **PanoseMatching** AS **cdrPanoseMatchingType**

Application

Description

The **PanoseMatching** property sets a value associated with a replacement font in CorelDRAW. When you open a file that contains a font that is not installed on your computer, PANOSE provides you with a list of fonts you can use to substitute the font. You can accept the suggestion, or choose another font. You can make the substitution for the current document only, or make it permanent.

Rather than substituting missing fonts each time you open a document containing missing fonts, you can set up a list of matches for uninstalled fonts. This list is saved for all subsequent documents when you exit CorelDRAW.

The **PanoseMatching** property returns a value of cdrPanoseMatchingType.

Example

The following code example specifies that a document opened with uninstalled system fonts permanently and automatically replaces the fonts with the closest font match available in the system:

```
Sub MatchingPanose()
PanoseMatching = cdrPanosePermanent
OpenDocument "C:\My Documents\Graphic1.cdr"
End Sub
```

{button ,AL(`CLS_Application;FNC_PanoseMatching')} **Related Topics**

cdrPanosePrompt=0

cdrPanoseTemporary=1

cdrPanosePermanent=2

Application.AddIns

Property **AddIns** AS **AddIns**

Application

_ENG Get Addins

Member of **Application**

Read-Only

Return value

Returns AddIns

Example

Example of usage goes here

{button ,AL(`CLS_Application;FNC_AddIns')} **Related Topics**

Document properties

Document Legend

- Active

- ActiveLayer

- ActivePage

- ActivePowerClip

- ActiveShape

- ActiveWindow

- Application
  - ApplyToDuplicate
  - CurvePrecision

- DataFields
  - Dirty
  - DrawingOriginX
  - DrawingOriginY
  - EditAcrossLayers

- FileName

- FilePath

- FullFileName

- Grid

- Index
  - Keywords
  - Name
  - Notes

- Pages

- Parent
  - PreserveSelection

- PDFSettings

- PrintSettings

- Properties
  - ReferencePoint
  - Resolution

- Rulers

- SelectionRange
  - ShapeEnumDirection

- Title
  - Unit

- Views

- Windows
  - WorldScale

Document methods

Document Legend

- Activate
- AddPages
- AddPagesEx
- BeginCommandGroup
- ClearSelection
- Close
- CreateView
- EndCommandGroup
- Export
- ExportBitmap
- ExportEx
- GetUserArea
- GetUserClick
- InsertPages
- InsertPagesEx
- PrintOut
- PublishToPDF
- Redo
- Repeat
- ResetSettings
- ResolveAllBitmapsLinks
- RestoreSettings
- Save
- SaveAs
- SaveSettings
- Selection
- Undo

Document events

Document

AfterPrint
AfterSave
AfterSaveAs
BeforeClose
BeforePrint
BeforeSave
BeforeSaveAs
LayerCreated
LayerDeleted
LayerSelected
PageCreated
PageDeleted
PageSelected
SelectionChanged
ShapeCreated
ShapeDeleted
ShapeMoved

Document.PDFSettings

Property **PDFSettings** As **PDFVBASettings**

Gets PDF settings for the specified document

Member of **Document**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Document;FNC_PDFSettings')} **Related Topics**

Document.PublishToPDF

Sub **PublishToPDF**(ByVal **FileName** As String)

Publishes the specified document to PDF

Member of **Document**

Parameters	Description
FileName	Description of FileName goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_Document;FNC_PublishToPDF')} **Related Topics**

Document

Class **Document**

Properties Methods Events Referenced by

The **Document** class defines the characteristics of Document objects and describes the look and behavior of the objects through its properties and methods. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is often referred to as a "page". You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

New documents use the preset options built into the CorelDRAW application. Every new document allows you to save settings so that new drawings opened in CorelDRAW open to the settings established in this Document page. These options include:

- Options set in the Options dialog box
- Toolbar settings
- Docker window
- Color palette

You can add, delete, and rename pages in documents as well as create web documents. Detailed information about each CorelDRAW document can be viewed and saved in a text file within a word processing application for future use.

{button ,AL(`CLS_Document')} **Related Topics**

Document.DataFields

Property **DataFields** As **DataFields**

Description

The **DataFields** property returns the **DataFields** collection of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A data item stores information about a shape object within the CorelDRAW development environment. By customizing a data item, you are customizing information about an object within CorelDRAW.

Before you assign data to objects in a drawing, you need to know what information you want to display. By default, CorelDRAW creates four data fields: Name, Cost, Comments, and CDRStaticID. The first three fields are for your convenience, and can be edited or deleted as required. The CDRStaticID field is used by CorelDRAW to identify objects, and can't be edited or deleted.Member of **Document**

The **DataFields** property returns a Read-Only value.

Example

The following code example counts the number of data fields in the active document of CorelDRAW:

```
Sub DocField()
MsgBox ActiveDocument.DataFields.Count
End Sub
```

{button ,AL(`CLS_Document;FNC_DataFields')} **Related Topics**

The Drawing Window contains a CorelDRAW drawing. You can draw anywhere in the Drawing Window, but only objects that appear on the Drawing Page (indicated by a rectangle with a drop shadow) print.

A Document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is often referred to as a page. You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

A document page is often referred to as a "page". You can access any page in a multi-page document by clicking on a page tab in the Navigator of the Drawing Window. Each page in a multi-page document is accessible through the Navigator, located in the bottom-left corner of the Drawing Window. The Navigator allows you to move through your document quickly and shows the total number of pages in your drawing and the number of the page currently displayed. You can move to a page in your document by clicking that page number in the Navigator.

Document.Application

Property **Application** AS **Application**

Document

Description

The **Application** property returns a value associated with the main CorelDRAW application. An application object is always available in VBA and its properties and methods are global, making it unnecessary to explicitly specify the Application object. For example, with the Application object, CorelScript and Application.CorelScript can be used interchangeably because they represent the same application object.

The **Application** property returns a Read-Only value.

Example

The following code example displays the current version of CorelDRAW in a message box:

```
Sub DocApp()
With ActiveDocument
    MsgBox .Application.Version
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Application')} **Related Topics**

Document.Parent

Property **Parent** AS **Documents**

Document

Description

The **Parent** property returns a value associated with the properties, methods, and controls of an object's parent in CorelDRAW's hierarchy of objects. A parent object acts as a container for child objects. For example, a form, as a container, is the parent object of a check box on that form.

Most objects have either a **Parent** property or a **Collection** property, pointing to the object's parent object in the object model. The **Collection** property is used if the parent object is a collection.

The **Parent** property returns a Read-Only value.

Example

The following code example displays the name of the active document's parent layer in a message box:

```
Sub DocParent()
With ActiveDocument
    MsgBox .Parent.ActiveLayer.Name
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Parent')} **Related Topics**

Document.Name

Property **Name** AS String

Document

Description

The **Name** property returns or sets the name of the current document in CorelDRAW. The name of the current document appears in the title bar of the main application window after the application name.

The **Name** property returns a String value.

Example

The following code example displays the name of the active document in a message box:

```
Sub DocName()
MsgBox ActiveDocument.Name
End Sub
```

{button ,AL(`CLS_Document;FNC_Name')} **Related Topics**

Document.SaveAs

Sub **SaveAs**(ByVal **FileName** AS String, ByRef Options AS **StructSaveAsOptions**)

Document

Description

The **SaveAs** method saves the current document, with options available, in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to use the **SaveAs** method, you must pass the **FileName** and **Options** values as parameters:

Parameters	Description
FileName	This parameter specifies the string value that names the document you are saving in CorelDRAW. It becomes the Name of the document.
Options	This parameter identifies the **Save** options of the document. This value is optional and the default value is Nothing (0).

Example

The following code example creates a new CorelDRAW document with the **CreateDocument** method, adds a colored ellipse to the active layer of the document, and saves the new document with a new name:

```
Sub DocumentCreate()
Dim doc As Document
Set doc = CreateDocument()
'creates a new CorelDRAW document
doc.ActiveLayer.CreateEllipse (0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
'creates an ellipse in the active layer with a uniform fill color
doc.SaveAs "My New Document"
End Sub
```

{button ,AL(`CLS_Document;FNC_SaveAs')} **Related Topics**

Document.Save

Sub **Save**()

Document

Description

The **Save** method saves the current document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example opens an existing CorelDRAW (CDR) document, adds a page to the document, creates a colored ellipse on the new page, and saves the entire document:

```
Sub DocumentOpen()
Dim doc As Document
Set doc = OpenDocument("C:\Flower.cdr")
With doc
    .AddPages (1)
    .ActiveLayer.CreateEllipse(0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
    .Save
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Save')} **Related Topics**

Document.Pages

Property **Pages** AS **Pages**

Document

Description

The **Pages** property returns the **Pages** collection of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **Pages** property returns a Read-Only value.

Example

The following code example displays the name of the first page in the Pages collection in a message box:

```
Sub PagesItem()
With ActiveDocument.Pages
    MsgBox .Item(1).Name
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Pages')} **Related Topics**

Document.ReferencePoint

Property **ReferencePoint** AS **cdrReferencePoint**

Document

Description

The **ReferencePoint** property returns or sets the reference point of every object in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The reference point defines a central point in the document for the positioning of objects in the document.

The **ReferencePoint** property returns cdrReferencePoint.

Example

The following code example sets the reference point of the document to the lower left corner and creates a new shape point with coordinates (1, 6), one inch above and 6 inches to the right of the reference point. A rectangle and ellipse are created, and a connector line is drawn between them, starting at the tenth snap point of the rectangle to the first snap point of the ellipse:

```
Sub PointStart()
Dim sp as New ShapePoint
Dim s1 As Shape, s2 As Shape, c As Shape
ActiveDocument.ReferencePoint = cdrBottomLeft
sp.PositionX = 1
sp.PositionY = 6
'****Creates a shapepoint with coordinates (1,6)
Set s1 = ActiveLayer.CreateRectangle(0, 0, 3, 3)
Set s2 = ActiveLayer.CreateEllipse(4, 4, 6, 1)
Set c = ActiveLayer.CreateConnector(s1.Points(10), s2.Points(1))
    c.Connector.StartPoint = s1.Points(10)
    c.Connector.EndPoint = s2.Points(1)
End Sub
```

{button ,AL(`CLS_Document;FNC_ReferencePoint')} **Related Topics**

Document.ApplyToDuplicate

Property **ApplyToDuplicate** AS Boolean

Document

Description

The **ApplyToDuplicate** property returns a True or False value that indicates if a transformation effect is applied to a duplicate object instead of the original object in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

CorelDRAW offers you an easy way to experiment with different transformations. If you want to see the effect of a transformation but keep the original intact, you can transform a copy of the object.

CorelDRAW creates a copy of the object as it is being transformed, leaving the original unaffected. If you decide that you'd rather keep the original, you can simply delete the copy.

It is important to note that this is a global state of the document. Changing this property may affect other script execution. You should always restore the state of **ApplyDuplicate** after using it.

Example

The following code example creates an ellipse, then moves it by 1" horizontally creating a copy of it:

```
Sub Test()
    Dim d As Document
    Dim s As Shape
    Dim b As Boolean
    Set d = ActiveDocument
    Set s = ActiveLayer.CreateEllipse2(4, 4, 2)
    b = d.ApplyToDuplicate ' store previous state
    d.ApplyToDuplicate = True
    s.Move 1, 0
    d.ApplyToDuplicate = b ' restore state
End Sub
```

{button ,AL(`CLS_Document;FNC_ApplyToDuplicate')} **Related Topics**

Document.ActivePage

Property **ActivePage** AS **Page**

Document

Description

The **ActivePage** property returns a value associated with the active page of the current document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

The **ActivePage** property returns a Read-Only value.

Example

The following code example changes the background color of the current page in the open document to red:

```
Sub PageActive()
ActivePage.Color.RGBAssign 255, 0, 0
'turns the background color red
End Sub
```

{button ,AL(`CLS_Document;FNC_ActivePage')} **Related Topics**

Document.ActiveLayer

Property **ActiveLayer** AS **Layer**

Document

Description

The **ActiveLayer** property returns a value associated with the active layer of the current document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

The **ActiveLayer** property returns a Read-Only value.

Example

The following code example creates a new CorelDRAW document with the **CreateDocument** method, adds a colored ellipse to the active layer of the document, and saves the new document with a new name:

```
Sub DocumentCreate()
Dim doc As Document
Set doc = CreateDocument()
'creates a new CorelDRAW document
doc.ActiveLayer.CreateEllipse (0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
'creates an ellipse in the active layer of the new document with a uniform fill color
doc.SaveAs "My New Document"
End Sub
```

{button ,AL(`CLS_Document;FNC_ActiveLayer')} **Related Topics**

Document.Windows

Property **Windows** AS **Windows**

Document

Description

The **Windows** property returns the **Windows** collection of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The **Windows** property returns a Read-Only value.

Example

The following code example displays the caption of the second window in the **Windows** collection in a message box:

```
Sub WindowsItem()
MsgBox ActiveDocument.Windows.Item(2).Caption
End Sub
```

{button ,AL(`CLS_Document;FNC_Windows')} **Related Topics**

Document.ActiveWindow

Property **ActiveWindow** AS **Window**

Document

Description

The **ActiveWindow** property returns a value associated with the active window of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A window is a container for a drawing in CorelDRAW and is often referred to as a Drawing Window. You can draw anywhere in the window, but only objects that appear on the Drawing Page will print. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs.

The **ActiveWindow** property returns a Read-Only value.

Example

The following code example closes the active window, if the view type of that window is the Enhanced View:

```
Sub WindowClose()
With ActiveDocument.ActiveWindow
    If .ActiveView.Type = cdrEnhancedView Then
    .Close
End If
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_ActiveWindow')} **Related Topics**

Document.Close

Sub **Close**()

Document

Description

The **Close** method closes the active document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example closes the active document in CorelDRAW:

```
Sub DocClose()
ActiveDocument.Close
End Sub
```

{button ,AL(`CLS_Document;FNC_Close')} **Related Topics**

Document.Undo

Sub **Undo**()

Document

Description

The **Undo** method reverses the last operation performed in the active document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example reverses the last operation in the active document of CorelDRAW:

```
Sub Reverse()
ActiveDocument.Undo
End Sub
```

{button ,AL(`CLS_Document;FNC_Undo')} **Related Topics**

Document.Redo

Sub **Redo**()

Document

Description

The **Redo** method reverses the last Undo operation performed in the active document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example reverses the last undo operation in the active document of CorelDRAW:

```
Sub Reverse()
ActiveDocument.Redo
End Sub
```

{button ,AL(`CLS_Document;FNC_Redo')} **Related Topics**

Document.Repeat

Sub **Repeat**()

Document

Description

The **Repeat** method repeats the last operations performed in the active document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example repeats the last operation in the active document of CorelDRAW:

```
Sub OnRepeat()
ActiveDocument.Repeat
End Sub
```

{button ,AL(`CLS_Document;FNC_Repeat')} **Related Topics**

Document.Activate

Sub **Activate**()

Document

Description

The **Activate** method opens a document in the main application window in CorelDRAW, if the window is not currently open, and makes the document active. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example creates two new documents in CorelDRAW. An ellipse is added to the first document and a rectangle is added to the second document. The first document is made the active document with the Activate method and a new page is added to the first document:

```
Sub DocumentActive()
CreateDocument
'creates the first document
ActiveLayer.CreateEllipse2 3, 2, 1
CreateDocument
'creates the second document
ActiveLayer.CreateRectangle 2, 3, 4, 5
Documents(1).Activate
'first document is the active document
ActiveDocument.AddPages 1
'you do not need to activate a document before working with it
'the last two lines of this procedure can be replaced with:
'Documents(1).AddPages 1
End Sub
```

{button , ALink(CLS_

Document;FNC_Activate)} **Related Topics**

Document.Unit

Property **Unit** AS **cdrUnit**

Document

Description

The **Unit** property returns or sets the unit of measurement for VBA commands in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **Unit** property returns cdrUnit.

Example

The following code example sets the document unit of measurement to inches:

```
Sub Measurement()
ActiveDocument.Unit = cdrInch
End Sub
```

{button ,AL(`CLS_Document;FNC_Unit')} **Related Topics**

Document.DrawingOriginX

Property **DrawingOriginX** AS Double

Document

Description

The **DrawingOriginX** property returns or sets the horizontal drawing origin, in document units, in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The rulers can help you determine the size and position of objects. Before using the rulers, determine the position of the ruler origin - the place where the rulers' 0 points intersect. Putting the ruler origin exactly where you want it on the Drawing Page ensures that the ruler coordinates begin from the exact location you need.

Example

The following code example sets the drawing origins in the active document:

```
Sub Draw()
ActiveDocument.DrawingOriginX = 2
ActiveDocument.DrawingOriginY = 2
End Sub
```

{button ,AL(`CLS_Document;FNC_DrawingOriginX')} **Related Topics**

Document.DrawingOriginY

Property **DrawingOriginY** AS Double

Document

Description

The **DrawingOriginY** property returns or sets the vertical drawing origin, in document units, in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The rulers can help you determine the size and position of objects. Before using the rulers, determine the position of the ruler origin - the place where the rulers' 0 points intersect. Putting the ruler origin exactly where you want it on the Drawing Page ensures that the ruler coordinates begin from the exact location you need.

Example

The following code example sets the drawing origins in the active document:

```
Sub Draw()
ActiveDocument.DrawingOriginX = 2
ActiveDocument.DrawingOriginY = 2
End Sub
```

{button ,AL(`CLS_Document;FNC_DrawingOriginY')} **Related Topics**

Document.AddPages

Function **AddPages**(ByVal **NumberOfPages** AS Long) AS **Page**

Document

Description

The **AddPages** method appends pages to a document in CorelDRAW. The value returned is the first page added. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **AddPages** method, you must pass the **NumberOfPages** value as a parameter:

Parameters	Description
NumberOfPages	This parameter identifies the number of pages to add to a document in CorelDRAW.

Example

The following code example opens an existing CorelDRAW (CDR) document, adds a page to the document, creates a colored ellipse on the new page, and saves the entire document:

```
Sub DocumentOpen()
Dim doc As Document
Set doc = OpenDocument("C:\Flower.cdr")
With doc
    .AddPages (1)
    .ActiveLayer.CreateEllipse(0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
    .Save
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_AddPages')} **Related Topics**

Document.InsertPages

Function **InsertPages**(ByVal **NumberOfPages** AS Long, ByVal **BeforePage** AS Boolean, ByVal **Page** AS Long) AS **Page**

Document

Description

The **InsertPages** method inserts pages between two other pages of a document in CorelDRAW. The value returned is the first page inserted. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **InsertPages** method, you must pass the several values as a parameters:

Parameters	Description
NumberOfPages	This parameter identifies the number of pages to add to a document in CorelDRAW.
BeforePage	Returns a True or False value indicating if the pages are inserted before the current page in the document.
Page	Lets you specify the page number from where to insert pages.

Example

The following code example inserts 4 pages after the current page, starting on the fifth page, in the active document of CorelDRAW:

```
Sub Insert()
ActiveDocument.InsertPages 0, 4, 5
End Sub
```

{button ,AL(`CLS_Document;FNC_InsertPages')} **Related Topics**

Document.Selection

Function **Selection**() AS **Shape**

Document

Description

The **Selection** property returns a shape in the document with a specified selection shape type in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example locks the objects in the current selection in the active document of CorelDRAW:

```
Sub Lock()
ActiveDocument.Selection.Locked = True
End Sub
```

{button ,AL(`CLS_Document;FNC_Selection')} **Related Topics**

Document.ClearSelection

Sub **ClearSelection**()

Document

Description

The **ClearSelection** method de-selects any objects that are selected in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example creates two separate ellipses with the **CreateEllipse** method. The active document is cleared, both ellipses are added to the active selection, and the ellipses are combined to create a single shape object. The intersections, or CrossPoints, of this new shape are identified by creating small circles around each point, using its x and y coordinates to serve as the coordinates of the small circles:

```
Sub XPosition()
Dim s1 As Shape, s2 As Shape
Dim s As Shape
Dim cps As CrossPoints
Dim cp As CrossPoint
Set s1 = ActiveLayer.CreateEllipse(4, 2, 1, 0)
Set s2 = ActiveLayer.CreateEllipse(3, 1, 2, 5)
    ActiveDocument.ClearSelection
    s1.AddToSelection
    s2.AddToSelection
    ActiveSelection.Combine
Set s = ActiveSelection.Shapes(1)
Set cps = s.Curve.Subpaths(1).GetIntersections(s.Curve.Subpaths(2))
For Each cp In cps
    ActiveLayer.CreateEllipse2 cp.PositionX, cp.PositionY, 0.1
Next cp
End Sub
```

{button ,AL(`CLS_Document;FNC_ClearSelection')} **Related Topics**

Document.Export

Sub **Export**(ByVal **FileName** AS String, ByVal **Filter** AS **cdrFilter**, ByVal Range AS **cdrExportRange**, ByRef Options AS **StructExportOptions**)

Document

Description

The **Export** method exports a document to a vector of bitmap file. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to export a document, you must specify several values as parameters:

Parameters	Description
FileName	This parameter specifies the string value that names the document you are saving in CorelDRAW. It becomes the Name of the document.
Filter	The **Filter** parameter returns or sets the filter that is used when saving objects in CorelDRAW. A filter is the name for an application that translates digital information from one form to another. Import/Export filters convert files from one format to another. When you select a file format in the Export dialog box of CorelDRAW, you automatically activate the appropriate filter application to perform the translation. The Filter parameter returns a value of cdrFilter.
Range	The **Range** parameter returns or sets the pages that are exported in CorelDRAW. You have the option of exporting all or several pages in a document, a single page in a document, or a selection of objects within a document. The **Range** parameter returns a value of cdrExportRange.
Options	This parameter identifies the **Save** options of the document. This value is optional and the default value is Nothing (0).

Example

The following code example creates a rectangle with fountain fill and exports it to a JPEG file:

```
Sub Test()
    Dim opt As New StructExportOptions
    Dim s As Shape
    ActiveDocument.Unit = cdrInch
    Set s = ActiveLayer.CreateRectangle(0, 0, 1, 1)
    s.Fill.ApplyFountainFill CreateColorEx(5005, 255, 0, 0), _
                CreateColorEx(5005, 0, 0, 0)
    opt.AntiAliasingType = cdrNormalAntiAliasing
    opt.ImageType = cdrRGBColorImage
    opt.Overwrite = True
    opt.ResolutionX = 72
    opt.ResolutionY = 72
    opt.SizeX = opt.ResolutionX * s.SizeWidth
    opt.SizeY = opt.ResolutionY * s.SizeHeight
    ActiveDocument.Export "C:\Rect.jpg", cdrJPEG, cdrSelection, opt
End Sub
```

{button ,AL(`CLS_Document;FNC_Export')} **Related Topics**

Document.ResolveAllBitmapsLinks

Sub **ResolveAllBitmapsLinks**()

Document

Description

The **ResolveAllBitmapLinks** method embeds all externally-linked bitmaps in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An embedded object is information from a file created in one application (the server application) that has been inserted into a file in another application (the client application). For example, you can embed a graphic created in Corel PHOTO-PAINT in CorelDRAW.

Example

The following code example embeds all the bitmaps into the active document of CorelDRAW:

```
Sub Test()
    ActiveDocument.ResolveAllBitmapLinks
End Sub
```

{button ,AL(`CLS_Document;FNC_ResolveAllBitmapsLinks')} **Related Topics**

Document.Dirty

Property **Dirty** AS Boolean

Document

Description

The **Dirty** property returns or sets a True or False value that allows users to determine if the document is dirty and if so, set the dirty flag in the document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This property determines whether the document was modified after it was last saved.

Example

The following code example closes the document unconditionally. Even if it is not saved, no warning is issued:

```
Sub Test()
    With ActiveDocument
        .Dirty = False
        .Close
    End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Dirty')} **Related Topics**

cdrCursorWinAppStarting	0
cdrCursorWinArrow	1
cdrCursorWinCross	2
cdrCursorWinHelp	3
cdrCursorWinIbeam	4
cdrCursorWinNo	5
cdrCursorWinSizeAll	6
cdrCursorWinSizeNeSw	7
cdrCursorWinSizeNs	8
cdrCursorWinSizeNwSe	9
cdrCursorWinSizeWe	10
cdrCursorWinUpArrow	11
cdrCursorWinWait	12
cdrCursorSmallcrosshair	1
cdrCursorPick	5
cdrCursorNodeEdit	6
cdrCursorEyeDrop	34
cdrCursorExtPick	60
cdrCursorPickNone	79
cdrCursorPenJoin	91
cdrCursorPickOvertarget	107
cdrCursorTrimSingle	139
cdrCursorWeldSingle	141
cdrCursorIntersectSingle	143

Document.GetUserClick

Function **GetUserClick**(ByRef **X** AS Double, ByRef **Y** AS Double, ByRef **ShiftState** AS Long, ByVal **TimeOut** AS Long, ByVal **Snap** AS Boolean, ByVal **CursorShape** AS **cdrCursorShape**) AS Long

Document

Description

The **GetUserClick** method returns or sets a user click position and information about the state of the CTRL, ALT and SHIFT keys at the moment of the click.

In order to use this method, you must pass several values as parameters:

Parameters	Description
X	Returns the X coordinate of the selection click.
Y	Returns the Y coordinate of the selection click.
ShiftState	Lets you specify the shift state.
TimeOut	Lets you specify the amount of time, in seconds, to wait for the user to click. Defaults to 10.
Snap	Lets you ignore or recognize the snap. This value is optional and defaults to True. 0 recognizes the snap, 1 ignores the snap.
CursorShape	Lets you specify the shape of the cursor. This value returns cdrCursorShape.

{button ,AL(`CLS_Document;FNC_GetUserClick')} **Related Topics**

Document.GetUserArea

Function **GetUserArea**(ByRef **x1** AS Double, ByRef **y1** AS Double, ByRef **x2** AS Double, ByRef **y2** AS Double, ByRef **ShiftState** AS Long, ByVal **TimeOut** AS Long, ByVal **Snap** AS Boolean, ByVal **CursorShape** AS **cdrCursorShape**) AS Long

Document

Description

The GetUserArea method returns or sets a user area and information about the state of the CTRL, ALT and SHIFT keys at the moment of a user click.

In order to use this method, you must pass several values as parameters:

Parameters	Description
x1	The x1 parameter defines the horizontal, or X, coordinate for the upper left corner of a rectangle shape on the active page that defines the user area. This value is measured in document units.
y1	The y1 parameter defines the vertical, or Y, coordinate for the upper left corner of a rectangle shape on the active page that defines the user area. This value is measured in document units..
x2	The x2 parameter defines the horizontal, or X, coordinate for the lower right corner of a rectangle shape on the active page that defines the user area. This value is measured in document units..
y2	The y2 parameter defines the vertical, or Y, coordinate for the lower right corner of a rectangle shape on the active page that defines the user area. This value is measured in document units..
ShiftState	Lets you specify the shift state.
TimeOut	Lets you specify the amount of time, in seconds, to wait for the user to click. Defaults to 10.
Snap	Lets you ignore or recognize the snap. This value is optional and defaults to True. 0 recognizes the snap, 1 ignores the snap.
CursorShape	Lets you specify the shape of the cursor. This value returns cdrCursorShape.

{button ,AL(`CLS_Document;FNC_GetUserArea')} **Related Topics**

Document.BeginCommandGroup

Sub **BeginCommandGroup**(ByVal CommandName AS String)

Document

Description

The **BeginCommandGroup** method starts a group of commands that have a clean undo stack in a document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

By starting a command group, you group a series of document handling commands that can undone in a single step. In order to use this method, you must pass the command name as a parameter:

Parameters	Description
CommandName	The **CommandName** specifies a string value that identifies a series of document commands in a document. This value is optional.

Example

The following code example creates a flower. After the macro finishes, its result can be undone in one step. The Edit menu fill display "Undo Flower" item:

```
Sub Test()
    Const Leaves As Long = 30
    Const Radius As Double = 2
    Dim stp As Double
    Dim i As Long
    Dim d As Document
    Dim s As Shape
    stp = 360 / Leaves
    Set d = ActiveDocument
    d.DrawingOriginX = 0
    d.DrawingOriginY = 0
    d.BeginCommandGroup "Flower"
    Set s = d.ActiveLayer.CreateEllipse2(0, 0, Radius)
    s.Fill.UniformColor.RGBAssign 100, 50, 50
    Set s = d.ActiveLayer.CreateEllipse2(Radius + 0.5, 0, 0.5, 0.25)
    s.Fill.UniformColor.RGBAssign 255, 255, 0
    s.RotationCenterX = 0
    s.RotationCenterY = 0
    d.ApplyToDuplicate = True
    For i = 1 To Leaves - 1
        s.Rotate i * stp
    Next i
    d.EndCommandGroup
End Sub
```

{button ,AL(`CLS_Document;FNC_BeginCommandGroup')} **Related Topics**

Document.EndCommandGroup

Sub **EndCommandGroup**()

Document

Description

The **EndCommandGroup** method ends a group of commands that have a clean undo stack in a document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

By starting a command group, you group a series of document handling commands that can undone in a single step. In order to use this method, you must pass the command name as a parameter:

Example

The following code example creates a flower. After the macro finishes, its result can be undone in one step. The Edit menu fill display "Undo Flower" item:

```
Sub Test()
    Const Leaves As Long = 30
    Const Radius As Double = 2
    Dim stp As Double
    Dim i As Long
    Dim d As Document
    Dim s As Shape
    stp = 360 / Leaves
    Set d = ActiveDocument
    d.DrawingOriginX = 0
    d.DrawingOriginY = 0
    d.BeginCommandGroup "Flower"
    Set s = d.ActiveLayer.CreateEllipse2(0, 0, Radius)
    s.Fill.UniformColor.RGBAssign 100, 50, 50
    Set s = d.ActiveLayer.CreateEllipse2(Radius + 0.5, 0, 0.5, 0.25)
    s.Fill.UniformColor.RGBAssign 255, 255, 0
    s.RotationCenterX = 0
    s.RotationCenterY = 0
    d.ApplyToDuplicate = True
    For i = 1 To Leaves - 1
        s.Rotate i * stp
    Next i
    d.EndCommandGroup
End Sub
```

{button ,AL(`CLS_Document;FNC_EndCommandGroup')} **Related Topics**

Document.FilePath

Property **FilePath** AS String

Document

Description

The **FilePath** property returns a string value associated with the computer location of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This value returns a Read-Only value.

Example

The following code example displays the file path of the active document in a message box:

```
Sub Path()
MsgBox ActiveDocument.FilePath
End Sub
```

{button ,AL(`CLS_Document;FNC_FilePath')} **Related Topics**

Document.FileName

Property **FileName** AS String

Document

Description

The **FileName** property returns a string value associated with the name of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This value returns a Read-Only value.

Example

The following code example displays the file name of the active document in a message box:

```
Sub Path()
MsgBox ActiveDocument.FileName
End Sub
```

{button ,AL(`CLS_Document;FNC_FileName')} **Related Topics**

Document.FullFileName

Property **FullFileName** AS String

Document

Description

The FullFileName property returns a string value associated with the file path and file name of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This value returns a Read-Only value.

Example

The following code example displays the file path and file name of the active document in a message box:

```
Sub Path()
MsgBox ActiveDocument.FullFileName
End Sub
```

{button ,AL(`CLS_Document;FNC_FullFileName')} **Related Topics**

Document.Resolution

Property **Resolution** AS Long

Document

Description

The **Resolution** property returns or sets the resolution of a document, measured in dots per inch (dpi), in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Resolution refers to the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size. The texture fill pattern is treated as a bitmap and its resolution determines the DPI value in the texture fill pattern.

DPI is a measure of a printer's resolution in dots per inch. Typical desktop laser printers print at 300 dpi. Image setters print at 1270 or 2540 dpi. Printers with higher dpi capabilities produce smoother and cleaner output. The term dpi is also used to measure scanning resolution and to indicate bitmap resolution.

Example

The following code example set the resolution of the active document to 400 dpi (dots per inch). The higher the resolution, the better your image will appear when printed:

```
Sub DocResolution()
With ActiveDocument
    .Resolution = 400
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Resolution')} **Related Topics**

cdrShapeEnumTopFirst=0

cdrShapeEnumBottomFirst=1

Document.ShapeEnumDirection

Property **ShapeEnumDirection** AS **cdrShapeEnumDirection**

Document

Description

The **ShapeEnumDirection** property returns or sets the shape direction in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This property controls how shapes in the **Shapes** collection are enumerated. By default (cdrShapeEnumTopFirst), the last created shape the index of 1 (i.e. accessible as Shapes(1)). If the value is set to cdrShapeEnumBottomFirst, the first created shape has the index of 1.

This property returns cdrShapeEnumDirection.

Example

The following code example creates an ellipse, then an artistic text object in a new document and applies a red fill to the ellipse (the bottommost object):

```
Sub Test()
    Dim d As Document
    Set d = CreateDocument
    d.ShapeEnumDirection = cdrShapeEnumBottomFirst
    d.ActiveLayer.CreateEllipse2 4, 4, 2
    d.ActiveLayer.CreateArtisticText 0, 0, "Text"
    d.ActiveLayer.Shapes(1).Fill.UniformColor.RGBAssign 255, 0, 0
End Sub
```

{button ,AL(`CLS_Document;FNC_ShapeEnumDirection')} **Related Topics**

Document.SelectionRange

Property **SelectionRange** AS **ShapeRange**

Document

Description

The **SelectionRange** property returns a value associated with a group of selected objects in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **SelectionRange** returns a Read-Only value.

Example

The following code example removes all text objects from the selection range in the active document:

```
Sub Test()
    Dim sr As ShapeRange
    Dim s As Shape
    Set sr = ActiveDocument.SelectionRange
    For Each s In sr
        If s.Type = cdrTextShape Then s.RemoveFromSelection
    Next s
End Sub
```

{button ,AL(`CLS_Document;FNC_SelectionRange')} **Related Topics**

Document.Rulers

Property **Rulers** AS **Rulers**

Document

Description

The **Rulers** property returns the **Rulers** collection of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A ruler is a measuring tool that displays on the left side and along the top of the Application Window. The rulers help you size and position the objects in your drawing.

The **Rulers** property returns a Read-Only value.

Example

The following code example sets the unit of measurement for the vertical rulers to inches in CorelDRAW:

```
Sub RulersVertical()
With ActiveDocument.Rulers
    .VUnits = cdrInch
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Rulers')} **Related Topics**

Document.Grid

Property **Grid** AS **Grid**

Document

Description

The **Grid** property returns a value associated with a grid of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A Grid is a series of evenly spaced horizontal and vertical dots that are used to help draw and arrange objects. You can use the controls on the Grid and Ruler Setup dialog box to set the grid's parameters. For greater accuracy, you can also have objects in your illustration snap to the grid when you move, or draw them.

The **Grid** property returns a Read-Only value.

Example

The following code example sets the Visible property of the grid to True, displaying the grid in the Drawing Window. The grid type pattern is set to dots and the SetFrequency method places 5 horizontal and vertical grid dots per square inch in the grid:

```
Sub GridType()
With ActiveDocument.Grid
    .Visible = True
    .Type = cdrGridDot
    .SetFrequency 5, 5
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Grid')} **Related Topics**

Document.Views

Property **Views** AS **Views**

Document

Description

The **Views** property returns the **Views** collection of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

The **Views** property returns a Read-Only value.

Example

The following code example displays the name of the second customized view in the **Views** collection in a message box:

```
Sub ViewsItem()
With ActiveDocument.Views
    MsgBox .Item(2).Name
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Views')} **Related Topics**

Document.CreateView

Function **CreateView**(ByVal **Name** AS String, ByVal **OriginX** AS Double, ByVal **OriginY** AS Double, ByVal Zoom AS Long, ByRef Page AS **Page**) AS **View**

Document

Description

The **CreateView** method creates a new view in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A view in CorelDRAW allows you to see your drawing according to your needs. You can use the view controls to change the way CorelDRAW displays objects, to magnify or reduce your view, or to save specific views for future use. You can use the default view name or create a unique name for each view that you create in CorelDRAW.

In order to use this method, you must pass several values as parameters:

Parameters	Description
Name	The **Name** parameter returns or sets a string value associated with the name of a view in CorelDRAW.
OriginX	The **OriginX** parameter returns or sets a value associated with the X coordinate of a view, measured in document units.
OriginY	The **OriginY** parameter returns or sets a value associated with the X coordinate of a view, measured in document units.
Zoom	The **Zoom** parameter returns or sets a user-defined magnification level for a customized view. This value is optional and the default value is 0.
Page	The **Page** parameter identifies the view to a specific page in the document. This value is optional and the default value is Nothing (0).

Example

The following code example creates a new view and activates it in the active document of CorelDRAW:

```
Sub Test()
    Dim v As View
    Dim x As Double, y As Double
    x = ActivePage.SizeWidth / 2
    y = ActivePage.SizeHeight / 2
    Set v = ActiveDocument.CreateView("My New View", x, y, 200)
    v.Activate
End Sub
```

{button ,AL(`CLS_Document;FNC_CreateView')} **Related Topics**

Document.ActivePowerClip

Property **ActivePowerClip** AS **PowerClip**

Document

Description

The **ActivePowerClip** property returns the current power clip object being edited in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Powerclip is a feature that lets you place objects (called contents objects) inside other objects (called container objects). If the contents object is larger than the container object, the contents object is automatically cropped. Only the contents that fit inside the container object are visible.

The **ActivePowerClip** property returns a Read-Only value.

Example

The following code example displays the number of shapes in the active power clip of the active document:

```
Sub PowerClip()
MsgBox ActiveDocument.ActivePowerClip.Shapes.Count
End Sub
```

{button ,AL(`CLS_Document;FNC_ActivePowerClip')} **Related Topics**

Document.WorldScale

Property **WorldScale** AS Double

Document

Description

The **WorldScale** property returns or sets the drawing scale in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The drawing scale represents the document world scale (e.g. 1:2, etc as set in Options>Document>Rulers>Edit Scale).

Example

The following code example sets the drawing scale to 2:1 and creates 2"x2" rectangle which is reported to be 1"x1" in CorelDRAW:

```
Sub Test()
    ActiveDocument.WorldScale = 0.5
    ActiveDocument.ActiveLayer.CreateRectangle 0, 0, 2, 2
End Sub
```

{button ,AL(`CLS_Document;FNC_WorldScale')} **Related Topics**

Document.PrintOut

Sub **PrintOut**()

Document

Description

The **PrintOut** method opens the Print dialog box in a document in CorelDRAW, and prints the document to the default printer using default settings. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example creates a rectangle in the active document and prints the document to the default system printer:

```
Sub Test()
    Dim s As Shape
    ActiveDocument.Unit = cdrInch
    ActiveLayer.CreateRectangle 0, 0, 1, 1
    ActiveDocument.PrintOut
End Sub
```

{button ,AL(`CLS_Document;FNC_PrintOut')} **Related Topics**

Document.ActiveShape

Property **ActiveShape** AS **Shape**

Document

Description

The **ActiveShape** property returns a value associated with the active shape object in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **ActiveShape** property returns a Read-Only value.

Example

The following code example displays the name of a data field, in an object data item, in the active document of CorelDRAW:

```
Sub FieldName()
With ActiveShape.ObjectData.Item(1)
    MsgBox .DataField.Name
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_ActiveShape')} **Related Topics**

Document.CurvePrecision

Property **CurvePrecision** AS Long

Document

Description

The **CurvePrecision** property returns a value associated with the precision value of a curve in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A curve object is a line, curve, or shape created with CorelDRAW's Freehand, Bezier, Artistic Media or Spiral tools. Rectangles. Ellipses, polygons, and text objects may be converted to curve objects by using the ConvertToCurves method of the Shape class.

The **CurvePrecision** property returns a Read-Only value.

Example

The following code example displays the curve precision value of the active document in a message box:

```
Sub Precision()
MsgBox ActiveDocument.CurvePrecision
End Sub
```

{button ,AL(`CLS_Document;FNC_CurvePrecision')} **Related Topics**

Document.AddPagesEx

Function **AddPagesEx**(ByVal **NumberOfPages** AS Long, ByVal **Width** AS Double, ByVal **Height** AS Double) AS **Page**

Document

Description

The **AddPagesEx** method adds pages, with a specified height and width, to a document in CorelDRAW The value returned is the first page inserted. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **AddPagesEx** method, you must pass the several values as a parameters:

Parameters	Description
NumberOfPages	This parameter identifies the number of pages to add to a document in CorelDRAW.
Width	This value specifies the width of the pages that are inserted in the document, measured in document units.
Height	This value specifies the width of the pages that are inserted in the document, measured in document units.

Example

The following code example opens an existing CorelDRAW (CDR) document, adds a page to the document that has a width and height of 10 document units, creates a colored ellipse on the new page, and saves the entire document:

```
Sub DocumentOpen()
Dim doc As Document
Set doc = OpenDocument("C:\Flower.cdr")
With doc
    .AddPagesEx 1, 10, 10
    .ActiveLayer.CreateEllipse(0, 3, 5, 1).Fill.UniformColor.CMYKAssign 0, 100, 100, 0
    .Save
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_AddPagesEx')} **Related Topics**

Document.InsertPagesEx

Function **InsertPagesEx**(ByVal **NumberOfPages** AS Long, ByVal **BeforePage** AS Boolean, ByVal **Page** AS Long, ByVal **Width** AS Double, ByVal **Height** AS Double) AS **Page**

Document

Description

The **InsertPagesEx** method inserts pages, with a specified height and width, to a document in CorelDRAW. The value returned is the first page inserted. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

In order to use the **InsertPagesEx** method, you must pass the several values as a parameters:

Parameters	Description
NumberOfPages	This parameter identifies the number of pages to add to a document in CorelDRAW.
BeforePage	Returns a True or False value indicating if the pages are inserted before the current page in the document.
Page	Lets you specify the page number from where to insert pages.
Width	This value specifies the width of the pages that are inserted in the document, measured in document units.
Height	This value specifies the width of the pages that are inserted in the document, measured in document units.

Example

The following code example inserts 4 pages after the current page (each with a width and height of 10 document units), starting on the fifth page, in the active document of CorelDRAW:

```
Sub Insert()
ActiveDocument.InsertPagesEx 0, 4, 5, 10, 10
End Sub
```

{button ,AL(`CLS_Document;FNC_InsertPagesEx')} **Related Topics**

Document.Title

Property **Title** AS String

Document

Description

The **Title** property returns a string value that identifies the title of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **Title** property returns a Read-Only value.

Example

The following code example displays the title of the active document in a message box:

```
Sub DocTitle()
MsgBox ActiveDocument.Title
End Sub
```

{button ,AL(`CLS_Document;FNC_Title')} **Related Topics**

Document.RestoreSettings

Sub **RestoreSettings**([ByVal Tag As String])

Description

The **RestoreSettings** method restores the document settings of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to use this method, you must pass the **Tag** value as a parameter:

Parameters	Description
Tag	The **Tag** parameter is a user-defined string value that describes the control. The user can change the tag description to suit their needs. This value is optional.

Example

The following code example restores the default settings of the active document in CorelDRAW:

```
Sub Restore()
ActiveDocument.RestoreSettings
End Sub
```

{button ,AL(`CLS_Document;FNC_RestoreSettings')} **Related Topics**

Document.SaveSettings

Sub **SaveSettings**([ByVal Tag As String])

Description

The **SaveSettings** method saves the current document settings in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to use this method, you must pass the **Tag** value as a parameter:

Parameters	Description
Tag	The **Tag** parameter is a user-defined string value that describes the control. The user can change the tag description to suit their needs. This value is optional.

Example

The following code example saves the document settings of the active document in CorelDRAW:

```
Sub Restore()
ActiveDocument.SaveSettings
End Sub
```

{button ,AL(`CLS_Document;FNC_SaveSettings')} **Related Topics**

Document.ResetSettings

Sub **ResetSettings**()

Description

The **ResetSettings** method clears the document settings in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document. Resets the document settings

Example

The following code example clears the settings of the active document in CorelDRAW:

```
Sub Restore()
ActiveDocument.ResetSettings
End Sub
```

{button ,AL(`CLS_Document;FNC_ResetSettings')} **Related Topics**

Document.Properties

Property **Properties** As **Properties**

Description

The **Properties** property returns a specified property of a document by referencing the index number of a property. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

The **Properties** property returns a Read-Only value.

Example

The following code example displays the number of properties associated with the active document in CorelDRAW:

```
Sub CountProp()
MsgBox ActiveDocument.Properties.Count
End Sub
```

{button ,AL(`CLS_Document;FNC_Properties')} **Related Topics**

Document.PrintSettings

Property **PrintSettings** As PrintSettings

Description

The **PrintSettings** property returns the print settings of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document. Get Print Settings

This value returns a Read-Only value.

{button ,AL(`CLS_Document;FNC_PrintSettings')} **Related Topics**

Document.PreserveSelection

Property **PreserveSelection** As Boolean

Description

The **PreserveSelection** property returns or sets a True or False value that indicates whether or not the current selection should always be preserved in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Example

The following code example preserves the current selection in the active document of CorelDRAW:

```
Sub Preserve()
ActiveDocument.PreserveSelection = True
End Sub
```

{button ,AL(`CLS_Document;FNC_PreserveSelection')} **Related Topics**

Document.Notes

Property **Notes** As String

Description

The **Notes** property returns or sets the notes of a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document. Gets or sets document notes

CorelDRAW offers advanced options that let you assign notes, keywords, and thumbnails so you can find your files more easily in future sessions.

Example

The following code example displays the notes associated with the active document in CorelDRAW:

```
Sub Note()
MsgBox ActiveDocument.Notes
End Sub
```

{button ,AL(`CLS_Document;FNC_Notes')} **Related Topics**

Document.Keywords

Property **Keywords** As String

Description

The **Keywords** property returns or sets the keywords associated with a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

CorelDRAW offers advanced options that let you assign notes, keywords, and thumbnails so you can find your files more easily in future sessions.

Example

The following code example displays the keywords associated with the active document in CorelDRAW:

```
Sub Note()
MsgBox ActiveDocument.Keywords
End Sub
```

{button ,AL(`CLS_Document;FNC_Keywords')} **Related Topics**

Document.EditAcrossLayers

Property **EditAcrossLayers** As Boolean

Description

The **EditAcrossLayers** property returns a True or False value that indicates if editing is available on multiple layers in the document of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

If you disable the Edit Across Layers function, you can work only on the active layer and the Desktop layer - you can't select or edit objects on inactive layers.

Example

The following code example allows you to edit objects on several different layers in the active document of CorelDRAW:

```
Sub Edit()
ActiveDocument.EditAcrossLayers = True
End Sub
```

{button ,AL(`CLS_Document;FNC_EditAcrossLayers')} **Related Topics**

Document.Active

Property **Active** AS Boolean

Document

Description

The **Active** property returns a True or False value indicating the active status of the current document in CorelDRAW. If the document is active, it is available for editing. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

This property returns a Read-Only value.

Example

The following code example closes the active document if it is currently active in CorelDRAW:

```
Sub DocActive()
If ActiveDocument.Active = True
    ActiveDocument.Close
End If
End Sub
```

{button ,AL(`CLS_Document;FNC_Active')} **Related Topics**

Document.Index

Property **Index** AS Long

Document

Description

The **Index** property returns a value associated with a document object in the **Documents** collection of CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

Documents(5) refers to the fifth document in the collection.

The **Index** property returns a Read-Only value.

Example

The following code example sets a new name for the first document in the **Documents** collection in CorelDRAW. The new document name displays in a message box:

```
Sub DocsItem()
With Documents
    .Item(1).Name = "CorelDRAW_Document"
End With
End Sub
```

{button ,AL(`CLS_Document;FNC_Index')} **Related Topics**

Document.ExportEx

Function **ExportEx**(ByVal **FileName** As String, ByVal **Filter** As **cdrFilter**, [ByVal Range As **cdrExportRange** = cdrCurrentPage (1)], [ByVal Options As **StructExportOptions** = 0], [ByVal PaletteOptions As **StructPaletteOptions**]) As Object

Document

Description

The **ExportEx** method exports a document, with specified options, to a vector of bitmap file. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to export a document, you must specify several values as parameters:

Parameters	Description
FileName	This parameter specifies the string value that names the document you are saving in CorelDRAW. It becomes the Name of the document.
Filter	The **Filter** parameter returns or sets the filter that is used when saving objects in CorelDRAW. A filter is the name for an application that translates digital information from one form to another. Import/Export filters convert files from one format to another. When you select a file format in the Export dialog box of CorelDRAW, you automatically activate the appropriate filter application to perform the translation. The Filter parameter returns a value of cdrFilter.
Range	The **Range** parameter returns or sets the pages that are exported in CorelDRAW. You have the option of exporting all or several pages in a document, a single page in a document, or a selection of objects within a document. The **Range** parameter returns a value of cdrExportRange.
Options	This parameter identifies the **Save** options of the exported document. This value is optional and the default value is Nothing (0).

Example

The following code example creates a rectangle with fountain fill and exports it to a JPEG file:

```
Sub Test()
    Dim opt As New StructExportOptions
    Dim s As Shape
    ActiveDocument.Unit = cdrInch
    Set s = ActiveLayer.CreateRectangle(0, 0, 1, 1)
    s.Fill.ApplyFountainFill CreateColorEx(5005, 255, 0, 0), _
                CreateColorEx(5005, 0, 0, 0)
    opt.AntiAliasingType = cdrNormalAntiAliasing
    opt.ImageType = cdrRGBColorImage
    opt.Overwrite = True
    opt.ResolutionX = 72
    opt.ResolutionY = 72
    opt.SizeX = opt.ResolutionX * s.SizeWidth
    opt.SizeY = opt.ResolutionY * s.SizeHeight
    ActiveDocument.ExportEx "C:\Rect.jpg", cdrJPEG, cdrSelection, opt
End Sub
```

{button ,AL(`CLS_Document;FNC_ExportEx')} **Related Topics**

Document.ExportBitmap

Function **ExportBitmap**(ByVal **FileName** As String, ByVal **Filter** As **cdrFilter**, [ByVal Range As **cdrExportRange** = cdrCurrentPage (1)], [ByVal ImageType As **cdrImageType** = cdrRGBColorImage (4)], [ByVal Width As Long = 0], [ByVal Height As Long = 0], [ByVal ResolutionX As Long = 72], [ByVal ResolutionY As Long = 72], [ByVal AntiAliasingType As **cdrAntiAliasingType** = cdrNormalAntiAliasing (1)], [ByVal Dithered As Boolean = False], [ByVal Transparent As Boolean = False], [ByVal UseColorProfile As Boolean = True], [ByVal Compression As **cdrCompressionType** = cdrCompressionNone (0)], [ByVal PaletteOptions As **StructPaletteOptions**]) As Object

Document

Description

The **ExportBitmap** method exports a bitmap in a document to a vector or bitmap file. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to export a document, you must specify several values as parameters:

Parameters	Description
FileName	This parameter specifies the string value that names the document you are saving in CorelDRAW. It becomes the Name of the document.
Filter	The **Filter** parameter returns or sets the filter that is used when saving objects in CorelDRAW. A filter is the name for an application that translates digital information from one form to another. Import/Export filters convert files from one format to another. When you select a file format in the Export dialog box of CorelDRAW, you automatically activate the appropriate filter application to perform the translation. The Filter parameter returns a value of cdrFilter.
Range	The **Range** parameter returns or sets the pages that are exported in CorelDRAW. You have the option of exporting all or several pages in a document, a single page in a document, or a selection of objects within a document. The **Range** parameter returns a value of cdrExportRange.
ImageType	The **ImageType** parameter identifies the image type being exported. This value is optional and returns cdrImageType. The default value is cdrRGBColorImage (4).
Width	This value sets the height of the bitmap, measured in document units.
Height	This value sets the height of the bitmap, measured in document units.
ResolutionX	The **ResolutionX** property returns or sets a numerical value that indicates the horizontal resolution level of an image that is exported in CorelDRAW.
ResolutionY	The **ResolutionY** property returns or sets a numerical value that indicates the vertical resolution level of an image that is exported in CorelDRAW.
AntiAliasingType	The **AntiAliasingType** property returns or sets a value that indicates the appearance of a bitmap image's edges when it is exported in CorelDRAW. Anti-aliasing is a method of smoothing the curved and diagonal edges contained in bitmap images. Anti-aliasing partially fills intermediate pixels along those edges to smooth the transition between the edge and the surrounding image. Anti-aliasing reduces or eliminates jagged edges. The **AntiAliasingType** property returns a value of cdrAntiAliasingType.
Dithered	Lets you specify if the image is dithered when exported. A dithered image uses color that is simulated by putting dots of another color very close together. Windows uses dithering to display colors that the graphics adapter can't display.
Transparent	Lets you specify the transparency of the image.
UseColorProfile	Lets you specify whether or not to use the color profile when exporting the image.

{button ,AL(`CLS_Document;FNC_ExportBitmap')} **Related Topics**

Document.BeforeSave

Event **BeforeSave**()

Document

Description

The **BeforeSave** event is triggered before the SaveDialog box appears when you are saving a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

{button ,AL(`CLS_Document;FNC_BeforeSave')} **Related Topics**

Document.AfterSave

Event **AfterSave**(ByVal **FileName** AS String)

Document

Description

The **AfterSave** event is triggered after a document is saved in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

In order to use this event, you must specify the **FileName** value as a parameter:

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

Parameters	Description
FileName	The **FileName** parameter identifies the file being saved in CorelDRAW. Once this file is saved, the **AfterSave** event is triggered and executed.

{button ,AL(`CLS_Document;FNC_AfterSave')} **Related Topics**

Document.BeforeClose

Event **BeforeClose**()

Document

Description

The **BeforeClose** event is triggered before a document is closed in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

{button ,AL(`CLS_Document;FNC_BeforeClose')} **Related Topics**

Document.PageSelected

Event **PageSelected**(ByRef **Page** AS **Page**)

Document

Description

The **PageSelected** event is triggered after a page is selected in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Page** value as a parameter:

Parameters	Description
Page	The **Page** parameter identifies the page that activates the **PageSelected** event when it is selected in a document.

{button ,AL(`CLS_Document;FNC_PageSelected')} **Related Topics**

Document.PageCreated

Event **PageCreated**(ByRef **Page** AS **Page**)

Document

Description

The **PageCreated** event is triggered after a page is created in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Page** value as a parameter:

Parameters	Description
Page	The **Page** parameter identifies the page that activates the **PageCreated** event when it is created in a document.

{button ,AL(`CLS_Document;FNC_PageCreated')} **Related Topics**

Document.LayerSelected

Event **LayerSelected**(ByRef **Layer** AS **Layer**)

Document

Description

The **LayerSelected** event is triggered after a layer is selected in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Layer** value as a parameter:

Parameters	Description
Layer	The **Layer** parameter identifies the layer that activates the **LayerSelected** event when it is selected in a document.

{button ,AL(`CLS_Document;FNC_LayerSelected')} **Related Topics**

Document.LayerCreated

Event **LayerCreated**(ByRef **Layer** AS **Layer**)

Document

Description

The **LayerCreated** event is triggered after a layer is created in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Layer** value as a parameter:

Parameters	Description
Layer	The **Layer** parameter identifies the layer that activates the **LayerCreated** event when it is created in a document.

{button ,AL(`CLS_Document;FNC_LayerCreated')} **Related Topics**

Document.ShapeCreated

Event **ShapeCreated**(ByRef **Shape** AS **Shape**)

Document

Description

The **ShapeCreated** event is triggered after a shape is selected in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Shape** value as a parameter:

Parameters	Description
Shape	The **Shape** parameter identifies the shape that activates the **ShapeCreated** event when it is created in a document.

{button ,AL(`CLS_Document;FNC_ShapeCreated')} **Related Topics**

Document.ShapeMoved

Event **ShapeMoved**(ByRef **Shape** AS **Shape**)

Document

Description

The **ShapeMoved** event is triggered when a shape object is moved in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Shape** value as a parameter:

Parameters	Description
Shape	The **Shape** parameter identifies the shape that activates the **ShapeCreated** event when it is created in a document.

{button ,AL(`CLS_Document;FNC_ShapeMoved')} **Related Topics**

Document.BeforePrint

Event **BeforePrint**()

Document

Description

The **BeforePrint** event is triggered before a document is printed in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

{button ,AL(`CLS_Document;FNC_BeforePrint')} **Related Topics**

Document.AfterPrint

Event **AfterPrint**()

Document

Description

The **AfterPrint** event is triggered after a document is printed in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

{button ,AL(`CLS_Document;FNC_AfterPrint')} **Related Topics**

Document.BeforeSaveAs

Event **BeforeSaveAs**()

Document

Description

The **BeforeSaveAs** event is triggered before a document is saved with the Save As feature in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

{button ,AL(`CLS_Document;FNC_BeforeSaveAs')} **Related Topics**

Document.AfterSaveAs

Event **AfterSaveAs**(ByVal **FileName** AS String)

Document

Description

The **AfterSaveAs** event is triggered after a document is saved with the Save As feature in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

Parameters	Description
FileName	The **FileName** parameter identifies the file being saved in CorelDRAW. Once this file is saved, the **AfterSaveAs** event is triggered and executed.

{button ,AL(`CLS_Document;FNC_AfterSaveAs')} **Related Topics**

Document.PageDeleted

Event **PageDeleted**(ByRef **Page** AS **Page**)

Document

Description

The **PageDeleted** event is triggered after a page is deleted in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A document page is sometimes called a page. A page contains all of the objects on a specific drawing page, which is the portion of the Drawing Window that appears on the printed page. This area is enclosed by a rectangle with a shadow effect. Although you can draw anywhere in the Drawing Window, only objects on the Drawing Page appear in your print jobs. You can access a document page in a multi-page document by clicking on a page tab at the bottom-left corner of the Drawing Window.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Page** value as a parameter:

Parameters	Description
Page	The **Page** parameter identifies the page that activates the **PageDeleted** event when it is deleted in a document.

{button ,AL(`CLS_Document;FNC_PageDeleted')} **Related Topics**

Document.LayerDeleted

Event **LayerDeleted**(ByRef **Layer** AS **Layer**)

Document

Description

The **LayerDeleted** event is triggered after a layer is deleted in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

A layer is one of a series of transparent planes on which you can place objects in a drawing. You can control how objects in your drawing overlay one another by moving the layer and the objects they contain. You can also lock layers and make them invisible and nonprintable. Use layers to help you organize different components of a complex drawing.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Layer** value as a parameter:

Parameters	Description
Layer	The **Layer** parameter identifies the layer that activates the **LayerDeleted** event when it is deleted in a document.

{button ,AL(`CLS_Document;FNC_LayerDeleted')} **Related Topics**

Document.ShapeDeleted

Event **ShapeDeleted**(ByRef **Shape** AS **Shape**)

Document

Description

The **ShapeDeleted** event is triggered after a shape object is deleted in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Shape** value as a parameter:

Parameters	Description
Shape	The **Shape** parameter identifies the shape that activates the **ShapeDeleted** event when it is deleted in a document.

{button ,AL(`CLS_Document;FNC_ShapeDeleted')} **Related Topics**

Document.SelectionChanged

Event **SelectionChanged**(ByRef **Shape** AS **Shape**)

Document

Description

The **SelectionChanged** event is triggered when another shape object is selected in a document in CorelDRAW. A document in CorelDRAW is single page drawing or a collection of single-page drawings that make up a multi-page document.

An event is an action recognized by an object, such as clicking the mouse or pressing a key, and for which you can write code to respond. Events can occur as a result of a user action or program code, or they can be triggered by the system.

In order to use this event, you must specify the **Shape** value as a parameter:

Parameters	Description
Shape	The **Shape** parameter identifies the shape that activates the **SelectionChanged** event when it is selected in a document.

{button ,AL(`CLS_Document;FNC_SelectionChanged')} **Related Topics**

CorelScript methods

CorelScript Legend

- AddArrowPoint
- AddBezierPoint
- AddEnvelopeEffect
- AddFreehandPoint
- AddNode
- AddObjectDataField
- AddPageFrame
- AddTabStop
- AfterObject
- AlignObjects
- AlignTextToBaseline
- AlignToCenterOfPage
- AlignToGrid
- AppendCurveLine
- AppendObjectToSelection
- ApplyBitmapEffect
- ApplyBlend
- ApplyContour
- ApplyDistortion
- ApplyDropShadow
- ApplyEnvelopeFrom
- ApplyExtrude
- ApplyFountainBitmapLens
- ApplyFountainFill
- ApplyFullColorFill
- ApplyLensEffect
- ApplyNoFill
- ApplyOutline
- ApplyPerspectiveEffect
- ApplyPostscriptFill
- ApplyPreset
- ApplyPresetEnvelope
- ApplyRotatedExtrude
- ApplyStyle
- ApplyTextureBitmapLens
- ApplyTextureFill
- ApplyTwoColorBitmapLens
- ApplyTwoColorFill
- ApplyUniformBitmapLens
- ApplyUniformFillColor
- BeforeObject
- BeginCommandGroup
- BeginDrawArrow
- BeginDrawBezier
- BeginDrawCurve
- BeginDrawFreehand
- BeginEditObject
- Benchmark
- BreakApart
- ChangeLayerColor
- ChangeLayerName

ClearAllObjectData
ClearEffect
ClearNodeSelection
ClearObjectData
ClickedDialogButton
CloneObject
CloseCurve
Combine
ConvertBitmapTo
ConvertColor
ConvertOutlineToObject
ConvertToBitmap
ConvertToCurves
CopyEffectFrom
CopyObjectDataFields
CopyPowerClip
CopyPropertiesFrom
CopyToClipboard
CopyToLayer
CreateAngleDimension
CreateArtisticText
CreateCallout
CreateConnector
CreateDimension
CreateEllipse
CreateGridBoxes
CreateGuidelineUsingAngle
CreateGuidelineUsingTwoPoints
CreateNewStyle
CreateObjectDataField
CreatePaletteFromDocument
CreatePaletteFromSelection
CreateRectangle
CreateSpiral
CreateSymPolygon
CreateTextString
CutToClipboard
DefineObjectDataField
DeleteGuidelineByIndex
DeleteGuidelineUsingAngle
DeleteGuidelineUsingTwoPoints
DeleteLayer
DeleteNode
DeleteObject
DeleteObjectDataField
DeletePages
DeletePaletteColor
DeleteStyle
DetachBlendPath
DisplayFacingPages
DistributeObjects
DrawCurveClosePath
DrawCurveCurveTo
DrawCurveLineTo
DrawCurveMoveTo

DropSymbol
DuplicateObject
EditAngleDimensionLabel
EditDimensionLabel
EditLayer
EditObjectCommand
EndCommandGroup
EndDrawArrow
EndDrawBezier
EndDrawCurve
EndDrawFreehand
EndEditObject
EndOfRecording
ExtractContents
ExtractText
FileClose
FileExit
FileExport
FileImport
FileNew
FileOpen
FilePrint
FileSave
FindNextObjectOfStyle
FindObjectOfStyle
FitTextToPath
FuseBlend
GetBitmapResolution
GetBitmapSize
GetCDRFileCompRatio
GetCDRFileKeywords
GetCDRFileLastSavedBy
GetCDRFileNotes
GetCDRFileThumbnail
GetCDRFileVersion
GetColor
GetCurrentPageName
GetCurrentPageOrientation
GetCurrentPageSize
GetCurrentPaletteName
GetCurrentWorkspaceName
GetCurveClose
GetCurveFirstNodePosition
GetCurveIthNodePosition
GetCurveLastNodePosition
GetCurveLength
GetCurveNodeCount
GetCurveSubpathCount
GetDocumentCount
GetDocumentName
GetEllipseClockwise
GetEllipseEndAngle
GetEllipseStartAngle
GetEllipseType
GetFillType

GetFountainFill
GetFountainFillColor
GetGuidelineInformation
GetNodeIndex
GetNodePosition
GetNodeSelectedCount
GetNodeType
GetNumberOfGuidelines
GetObjectCount
GetObjectData
GetObjectID
GetObjectsCDRStaticID
GetObjectType
GetOutline
GetOutlineColor
GetPageCount
GetPageSize
GetPaletteColor
GetPaletteColorCount
GetPaletteColorName
GetPolygonSharpness
GetPolygonSides
GetPolygonType
GetPosition
GetRectangleRadius
GetSegmentLength
GetSegmentType
GetSize
GetTextFontName
GetTextFontSize
GetTextString
GetTextWordCount
GetUniformFillColor
GetUserClick
GetUserDataField
GetWorkspaceCount
GetWorkspaceDescription
GetWorkspaceName
Group
InflateBitmap
InitBezierTool
InsertOLEObject
InsertOLEObjectFromFile
InsertPages
InsertPaletteColor
Intersection
IsBitmapExternallyLinked
IsDefaultWorkspace
IsDocument
IsSelection
LoadPalette
LoadStyles
LockGuidelineByIndex
MenuCommand
MergeBackText

MoveBezierControl
MoveCenter
MoveGuidelineUsingAngleByIndex
MoveGuidelineUsingTwoPointsByIndex
MoveLayerTo
MoveNode
MoveObject
MoveToLayer
NewLayer
OLEObjectDoVerb
OrderBackOne
OrderForwardOne
OrderObjectDataFields
OrderReverseOrder
OrderToBack
OrderToFront
OverprintFill
OverprintOutline
PasteCustomClipboardFormat
PasteFromClipboard
PasteObjectData
PasteSystemClipboardFormat
PlaceInside
RecorderApplyPerspective
RecorderBeginEditParaText
RecorderBeginEditText
RecorderEditParaReplaceText
RecorderEditParaTextChangeCase
RecorderEditParaTextCharAttributes
RecorderEditParaTextIndents
RecorderEditParaTextSpacing
RecorderEditReplaceText
RecorderEditTextChangeCase
RecorderEditTextCharAttributes
RecorderEndEditParaText
RecorderEndEditText
RecorderObjectScaleInfo
RecorderSelectObjectByIndex
RecorderSelectObjectsByIndex
RecorderSelectPreselectedObjects
RecorderStorePreselectedObjects
Redo
RedrawAllScreens
RedrawScreen
RegisterObject
RemoveAllGuidelines
RemoveFountainFillColor
RenameObjectDataField
RenameStyle
Repeat
RepeatLastCommand
ReplacePaletteColor
ResetTransfo
ResolveAllBitmapsLink
ResolveBitmapLink

ResumePainting
RevertToStyle
RotateObject
SavePalette
SaveStyleAs
SaveStyleProp
SaveTemplate
SelectAllObjects
SelectLayer
SelectNextNode
SelectNextObject
SelectNode
SelectNodeAt
SelectObjectAtPoint
SelectObjectOfCDRStaticID
SelectObjectOfType
SelectObjectsInRect
SelectPreviousObject
Separate
SetApplyToDuplicate
SetArtisticText
SetBullet
SetCharacterAttributes
SetColorOverride
SetCornerRoundness
SetCurrentDocument
SetCurrentPage
SetCurrentPageName
SetCurrentPageOrientation
SetCurrentPageSize
SetCurrentWorkspace
SetDocVisible
SetEllipseProperties
SetErrorHandling
SetFrameColumn
SetFullScreenPreview
SetIndents
SetLayerLocked
SetLayerPrintable
SetLayerVisible
SetMultiLayer
SetNodeType
SetObjectData
SetOptionsForAllPages
SetOutlineArrow
SetOutlineColor
SetOutlineMiscProperties
SetOutlineWidth
SetPageLayout
SetPageOrientation
SetPageSize
SetPageSizeFromPrinter
SetPaperColor
SetParagraphSpacing
SetPolygonProperties

SetPosition
SetReferencePoint
SetSegmentType
SetSize
SetTextString
SetToMasterLayer
SetUserDataField
SetVisible
ShareExtrudeVP
ShowPageBorder
SkewObject
SplitBlend
StartEditContents
StartOfRecording
StopEditContents
StoreColor
StraightenText
StretchObject
SuppressPainting
Trim
Undo
Ungroup
UngroupAll
UnlockGuidelineByIndex
UnRegisterObject
UnSelectAll
UpdateBitmapLink
Weld
ZoomIn
ZoomOut
ZoomToAllObjects
ZoomToHeight
ZoomToPage
ZoomToRectangle
ZoomToSelection
ZoomToWidth

CorelScript

Class **CorelScript**

Methods Referenced by

The **CorelScript** class defines the characteristics of Corel Script objects and describes the look and behavior of the objects through its properties and methods.

Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Click for a list of Corel applications that support Corel SCRIPT.

Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. If you run a script for an application that is not running, Corel SCRIPT automatically starts the application.

A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

The Corel SCRIPT programming language is based on the BASIC programming language. If you're already familiar with a version of BASIC, you'll find the Corel SCRIPT programming language easy to read and understand.

A script is a computer program that executes a series of instructions with a single command. Generally, scripts are used to automate repetitive tasks, or simplify complicated actions; they can also be used to prompt for user input, display messages, and interact with other applications.

Scripts can significantly increase your productivity with Corel applications such as CorelDRAW or VENTURA, by automating repetitive tasks. For example, a script could be used to open a group of files, perform a set of editing actions, or set an application's default properties. In their simplest form, scripts replicate a Corel application's keystrokes, and toolbar, menu, and mouse commands. In a more complex form, scripts can include the commands and constructs of a programming language. For example, you could create a script that only replicates an application's commands once a series of logical requirements have been met.

{button ,AL(`CLS_CorelScript')} **Related Topics**

CorelScript.SetVisible

Function **SetVisible**(ByVal **Visible** AS Boolean) AS Long

CorelScript

Description

This command makes the CorelDRAW Script Edtior visible.

Parameter	Description
.**Visible**	Set to TRUE (-1) to show the CorelDRAW Script Editor. Set to False (0) to hide the CorelDRAW Script Editor.

Example

```
.SetVisible -1
```

The above example makes the CorelDRAW Script Editor visible.

{button ,AL(`CLS_CorelScript;FNC_SetVisible')} **Related Topics**

CorelScript.FileOpen

Function **FileOpen**(ByVal **FileName** AS String) AS Long

CorelScript

Description

This command loads a drawing or Styles template into CorelDRAW.

Parameter	Description
.**FileName**	Lets you specify the name of the file to open.

Note

- You cannot change the active CorelDRAW document in a script except by using the .FileNew or .FileOpen command. Changing the active CorelDRAW document with keyboard and mouse actions does not affect an executing script.

Example

```
.FileOpen "C:\TEST1.CDR"
```

The above example opens a CorelDRAW file named "TEST1.CDR".

{button ,AL(`CLS_CorelScript;FNC_FileOpen')} **Related Topics**

CorelScript.FilePrint

Function **FilePrint**() AS Long

CorelScript

Description

This command prints the active document.

Example

```
.FilePrint
```

The above example sends the active document to the printer.

{button ,AL(`CLS_CorelScript;FNC_FilePrint')} **Related Topics**

CorelScript.FileImport

Function **FileImport**(ByVal **FileName** AS String, ByVal **FilterId** AS Long, ByVal **MaintainLayersAndPages** AS Boolean) AS Long

CorelScript

Description

This command brings graphics into CorelDRAW from other programs.

Parameter	Description
.**FileName**	Lets you specify the name of the file to import.
.**FilterID**	Lets you specify the type of file filter: 769 = Windows Bitmap (BMP) 770 = Paintbrush (PCX) 771 = Targa Bitmap (TGA) 772 = TIFF Bitmap (TIF) 773 = CompuServe Bitmap (GIF) 774 = JPEG Bitmap (JPG) 775 = Kodak Photo CD Image (PCD) 776 = Scitex CT Bitmap (SCT) 784 = Windows 3.x/NT Cursor resource (CUR) 785 = Windows 3.x/NT Icon resource (ICO) 786 = Windows 3.x/NT Bitmaps resource (EXE) 787 = GEM Paint File (IMG) 788 = Adobe Photoshop (PSD) 789 = Picture Publisher 4 (PP4) 791 = MACPaint Bitmap (MAC) 792 = OS/2 Bitmap (BMP) 793 = Wavelet Compressed Bitmaps (WVL) 800 = CALS Compressed Bitmap (CAL) 802 = Portable Network Graphics (PNG) 803 = Picture Publisher 5.0, 6 (PP5, PP6) 806 = Kodak FlashPix Image (FPX) 1280 = Computer Graphics Metafile (CGM) 1281 = HPGL Plotter File (PLT) 1282 = Micrografx 2.x, 3.x (DRW) 1284 = GEM File (GEM) 1285 = IBM PIF (PIF) 1287 = WordPerfect Graphics (WPG) 1289 = Encapsulated PostScript (EPS) 1290 = PostScript Interpreted (PS, PRN, ES) 1291 = MET Maetfile (MET) 1292 = NAP Metafile (NAP) 1293 = Macintosh Pict (PCT) 1294 = Windows Metafile (WMF) 1296 = AutoCad (DXF) 1298 = Scodl (SCD) 1300 = Enhanced Windows Metafile (EMF) 1302 = True Type Font (TTF) 1303 = Adobe Type 1 Font (PFB) 1305 = Adobe Illustrator (AI) 1312 = Corel Barista (HTM) 1313 = Corel Image Map (HTM) 1314 = Placeable Enhanced PDF (PDF) 1315 = Visio (VSD) 1329 = Frame Vector Metafile (FMV) 1333 = Adobe Portable Document File (PDF) 1334 = Lotus Pic (PIC) 1339 = Micrografx Designer 6.0 (DSF) 1557 = HyperText Markup Language (HTM) 1792 = Corel PHOTO-PAINT Image (CPT) Ver 5.0/6.0 1793 = Corel Presentation Exchange 6/7 (CMX) 1794 = Corel Presentation Exchange 5.0 (CMX) 1796 = CorelDRAW Compressed (CDX) 1797 = Corel CMX Compressed (CPX) 1799 = Corel PHOTO-PAINT Image (CPT) 2048 = ANSI Text (TXT) 2049 = MS Word for Windows 6/7 (DOC) 2050 = MS Word for Windows 2.x (DOC)

2051 = MS Word 4.0, 5.0, 5.5 (DOC)
2052 = MS Word for Macintosh 4.0, 5.0 (DOC)
2053 = Rich Text Format (RTF)
2055 = Corel WordPerfect 6/7/8 (WPD)
2056 = Corel WordPerfect 5.1 (WP5)
2057 = Corel Word Perfect 5.0 (WP5)
2058 = Corel Word Perfect 4.2 (WP5)
2059 = Word Star for Windows 1.x, 2.0 (WSW)
2060 = Word Star 7.0 (WSD)
2061 = Word Star 2000 (WSD)
2062 = XYWrite for Windows (XY)
2063 = Ami Professional 2.0, 3.0 (SAM)
2068 = MS Word 97 (DOC)

MaintainLayersAndPages When set to TRUE (-1) maintain layers and pages during import.

Example

```
.FileNew
.FileImport "C:\TEST1.BMP", 769, FALSE
```

The above example creates a new document and imports a Windows bitmap file named "TEST1.BMP" into the document.

{button ,AL(`CLS_CorelScript;FNC_FileImport')} **Related Topics**

CorelScript.FileExport

Function **FileExport**(ByVal **FileName** AS String, ByVal **FilterId** AS Long, ByVal **Width** AS Long, ByVal **Height** AS Long, ByVal **XResolution** AS Long, ByVal **YResolution** AS Long, ByVal **ImageType** AS Long, ByVal **AntiAliasing** AS Long, ByVal **Overwrite** AS Boolean, ByVal **SelectionOnly** AS Boolean) AS Long

0 CorelScript

Description

This command saves the current drawing in a format that other programs can read.

Parameter	Description
.**FileName**	Lets you specify the name of the file to export.
0	.**FilterID** Lets you specify the type of file filter.
1	769 = Windows Bitmap (BMP) 770 = Paintbrush (PCX) 771 = Targa Bitmap (TGA) 772 = TIFF Bitmap (TIF) 773 = CompuServe Bitmap (GIF) 774 = JPEG Bitmap (JPG) 776 = Scitex CT Bitmap (SCT) 787 = GEM Paint File (IMG) 788 = Adobe Photoshop (PSD) 791 = MACPaint Bitmap (MAC) 792 = OS/2 Bitmap (BMP) 800 = CALS Compressed Bitmap (CAL) 802 = Portable Network Graphics (PNG) 806 = Kodak FlashPix Image (FPX) 1280 = Computer Graphics Metafile (CGM) 1281 = HPGL Plotter File (PLT) 1282 = Micrografx 2.x, 3.x (DRW) 1284 = GEM File (GEM) 1285 = IBM PIF (PIF) 1287 = WordPerfect Graphics (WPG) 1289 = Encapsulated PostScript (EPS) 1293 = Macintosh Pict (PCT) 1294 = Windows Metafile (WMF) 1296 = AutoCad (DXF) 1298 = Scodl (SCD) 1300 = Enhanced Windows Metafile (EMF) 1302 = True Type Font (TTF) 1303 = Adobe Type 1 Font (PFB) 1305 = Adobe Illustrator (AI) 1312 = Corel Barista (HTM) 1313 = Corel Image Map (HTM) 1329 = Frame Vector Metafile (FMV) 1333 = Adobe Portable Document File (PDF) 1792 = Corel PHOTO-PAINT Image (CPT) Ver 5.0/6.0 1793 = Corel Presentation Exchange 6/7 (CMX) 1794 = Corel Presentation Exchange 5.0 (CMX) 1799 = Corel PHOTO-PAINT Image (CPT) 2048 = ANSI Text (TXT) 2049 = MS Word for Windows 6/7 (DOC) 2050 = MS Word for Windows 2.x (DOC) 2051 = MS Word 4.0, 5.0, 5.5 (DOC) 2052 = MS Word for Macintosh 4.0, 5.0 (DOC) 2053 = Rich Text Format (RTF) 2055 = Corel WordPerfect 6/7/8 (WPD) 2056 = Corel WordPerfect 5.1 (WP5) 2057 = Corel Word Perfect 5.0 (WP5) 2058 = Corel Word Perfect 4.2 (WP5) 2059 = Word Star for Windows 1.x, 2.0 (WSW) 2060 = Word Star 7.0 (WSD) 2061 = Word Star 2000 (WSD) 2062 = XYWrite for Windows (XY) 2068 = MS Word 97 (DOC)
2	.**Width** Lets you specify the width of the image in pixels.
3	.**Height** Lets you specify the height of the image in pixels.

4 .**XResolution** Lets you specify the horizontal resolution of the image in dots per inch (dpi).
5 .**YResolution** Lets you specify the vertical resolution of the image in dots per inch (dpi).
6 .**ImageType** Lets you specify the image type.
1 = Monochrome bitmap
3 = 8-bit paletted color bitmap
4 = 24-bit RGB color bitmap
6 = 32-bit CMYK bitmap
10 = 4-bit, 16 colors (standard VGA palette)
7 .**Antialiasing** 0 = None
1 = Normal
2 = Super-Sampling
8 .**Overwrite** When set to TRUE (-1) overwrites the file if one exists.
9 .**SelectionOnly** When set to TRUE (-1) exports only the current selection.

0 Example

```
.FileExport "C:\TEMP1.BMP", 769, 320, 400, 72, 72, 4
```

The above example exports a CorelDRAW file to a Windows bitmap named "TEMP1.BMP".

{button ,AL(`CLS_CorelScript;FNC_FileExport')} **Related Topics**

CorelScript.FileNew

Function **FileNew**() AS Long

0 CorelScript

Description

This command creates a new drawing.

Return Value

Returns one of the following values:

- TRUE (-1) – the file was created
- FALSE (0) – the file was not created

Note

- You cannot change the active CorelDRAW document in a script except by using the .FileNew or .FileOpen command. Changing the active CorelDRAW document with keyboard and mouse actions does not affect an executing script.

Example

```
.FileNew
```

The above example creates a new CorelDRAW document.

{button ,AL(`CLS_CorelScript;FNC_FileNew')} **Related Topics**

CorelScript.SetFullScreenPreview

Function **SetFullScreenPreview**(ByVal **FullScreen** AS Boolean) AS Long

0 CorelScript

Description

This command removes everything but your drawing from the screen. You cannot edit your drawing in this mode.

Parameter	Description
.**FullScreen**	Set to TRUE (-1) to remove everything but your drawing from the screen. Set to FALSE (0) to return to normal mode.

Example

```
.SetFullScreenPreview -1
```

The above example displays a full-screen preview of the active image.

{button ,AL(`CLS_CorelScript;FNC_SetFullScreenPreview')} **Related Topics**

CorelScript.SetCurrentPage

Function **SetCurrentPage**(ByVal **CurrentPage** AS Long) AS Long

0 CorelScript

Description

This command makes the specified page the current page.

Parameter	Description
.**CurrentPage**	Lets you specify which page to make the current page.

Example

```
.SetCurrentPage 2
```

The above example sets the second page as the current page.

{button ,AL(`CLS_CorelScript;FNC_SetCurrentPage')} **Related Topics**

CorelScript.Undo

Function **Undo**() AS Long

0 CorelScript

Description

This command reverses actions performed during the current session. Use Undo after you have made a change you do not want to implement. Immediately after you select .Undo, the .Redo command becomes available, allowing you to restore what you just undid. You cannot undo the following operations: any change of view (e.g., Zoom-in or Zoom-out); any file operations (e.g., Open, Save, or Import); any selection operations (e.g., Marquee select or Node select).

Example

```
.Undo
```

The above example undoes the last command.

{button ,AL(`CLS_CorelScript;FNC_Undo')} **Related Topics**

CorelScript.Redo

Function **Redo**() AS Long

0 CorelScript

Description

This command restores changes reversed by the Undo command. Redo becomes available immediately after you select the Undo command.

Example

```
.Redo
```

The above command reverses the last .Undo command and reinstates the previous deletion or reversal of actions.

{button ,AL(`CLS_CorelScript;FNC_Redo')} **Related Topics**

CorelScript.Repeat

Function **Repeat**() AS Long

0 CorelScript

Description

This command applies, if possible, the most recent command or action to selected object.

Example

```
.Repeat
```

The above example repeats the last command.

{button ,AL(`CLS_CorelScript;FNC_Repeat')} **Related Topics**

CorelScript.RepeatLastCommand

Function **RepeatLastCommand**() AS Long

0 CorelScript

Description

This command applies, if possible, the last command or action to selected object.

Example

```
.RepeatLastCommand
```

The above example repeats the last command.

{button ,AL(`CLS_CorelScript;FNC_RepeatLastCommand')} **Related Topics**

CorelScript.SelectAllObjects

Function **SelectAllObjects**() AS Long

0 CorelScript

Description

This command selects every object in your drawing, including any not currently in view.

Example

```
.SelectAllObjects
```

The above example selects all objects in the active document.

{button ,AL(`CLS_CorelScript;FNC_SelectAllObjects')} **Related Topics**

CorelScript.SetPosition

Function **SetPosition**(ByVal **XPos** AS Long, ByVal **YPos** AS Long) AS Long

0 CorelScript

Description

This command sets the position for placement of the selected object

Parameter	Description
.**XPos**	Lets you specify the X-coordinate of the new position in tenths of a micron.
0 .**YPos**	Lets you specify the Y-coordinate of the new position in tenths of a micron.

Example

```
.CreateRectangle 1350000, -1000000, 750000, -500000, 0
.CreateArtisticText "1"
.SetPosition -950000, 1250000
```

The above example creates a rectangle and positions a number '1' in its upper-left corner.

{button ,AL(`CLS_CorelScript;FNC_SetPosition')} **Related Topics**

CorelScript.GetPosition

Function **GetPosition**(ByRef **XPos** AS Long, ByRef **YPos** AS Long) AS Long

0 CorelScript

Description

This function returns the position coordinates of a selected object's reference point. If more than one object is selected, the function returns the position coordinates of the last selected object.

Parameter	Description
.**XPos**	Returns the X-coordinate of the selected object's reference point in tenths of a micron, relative to the center of the page. An object's default reference point is the upper-left corner.
0 .**YPos**	Returns the Y-coordinate of the selected object's reference point in tenths of a micron, relative to the center of the page. An object's default reference point is the upper-left corner.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100005, 0
id& = .GetObjectsCDRStaticID()
.GetPosition XPos&, YPos&
MESSAGE "Horizontal"+STR(XPos&)
MESSAGE "Vertical"+STR(YPos&)
```

The above example creates a rectangle then displays the coordinates of the upper-left corner in message boxes.

```
.CreateRectangle 1000000, 750000, 500000, 100005, 0
id& = .GetObjectsCDRStaticID()
.SetReferencePoint 3
.GetPosition XPos&, YPos&
MESSAGE "Horizontal"+STR(XPos&)
MESSAGE "Vertical"+STR(YPos&)
```

The above example creates a rectangle then displays the coordinates of the upper-right corner in message boxes. The upper-right coordinates are used because the selected object's reference point was changed with the .SetReferencePoint command.

{button ,AL(`CLS_CorelScript;FNC_GetPosition')} **Related Topics**

CorelScript.MoveObject

Function **MoveObject**(ByVal **XDelta** AS Long, ByVal **YDelta** AS Long) AS Long

0 CorelScript

Description

This command repositions the selected object to the specified location.

Parameter	Description
.**XDelta**	Lets you specify the distance the object is to be moved along the X-axis in tenths of a micron.
0 .**YDelta**	Lets you specify the distance the object is to be moved along the Y-axis in tenths of a micron.

Example

```
.SetPageSize 2159000, 2794000
.CreateRectangle 500000, -750000, -500000, 750000, 0
.MoveObject 250000, -750000
```

The above example creates a rectangle, then moves it to the bottom right corner of an 8.5 by 11 inch page.

{button ,AL(`CLS_CorelScript;FNC_MoveObject')} **Related Topics**

CorelScript.GetSize

Function **GetSize**(ByRef **XSize** AS Long, ByRef **YSize** AS Long) AS Long

0 CorelScript

Description

This function returns the size attributes of a selected object. If more than one object is selected, the function returns the size attributes of the last selected object.

Parameter	Description
.**XSize**	Returns the horizontal size of the selected object, in tenths of a micron.
0 .**YSize**	Returns the vertical size of the selected object, in tenths of a micron.

Example

```
.CreateRectangle 1000000, 750000, 450000, 100000, 0
id& = .GetObjectsCDRStaticID()
.GetSize XSize&, YSize&
MESSAGE "Horizontal"+STR(XSize&)
MESSAGE "Vertical"+STR(YSize&)
```

The above example returns the size of the selected rectangle and displays the width and height (in tenths of a micron) in message boxes.

{button ,AL(`CLS_CorelScript;FNC_GetSize')} **Related Topics**

CorelScript.SetSize

Function **SetSize**(ByVal **XSize** AS Long, ByVal **YSize** AS Long) AS Long

0 CorelScript

Description

This command lets you scale, mirror, or set the size of the selected object.

Parameter	Description
.**XSize**	Lets you specify the new horizontal size of the selected object, in tenths of a micron.
0 .**YSize**	Lets you specify the new vertical size of the selected object, in tenths of a micron.

Note

- To mirror an object, use negative values for the .XSize and .YSize parameters.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
id& = .GetObjectsCDRStaticID()
status& = .GetSize (XSize&, YSize&)
.SelectObjectOfCDRStaticID id&
.SetSize 2*XSize&, 3*YSize&
```

The above example gets the size of the selected rectangle and sets the width to twice the original size, and the height to three times the original size.

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
id& = .GetObjectsCDRStaticID()
status& = .GetSize (XSize&, YSize&)
.SelectObjectOfCDRStaticID id&
.SetSize -XSize&, YSize&
```

The above example horizontally mirrors the selected object, maintaining its original size.

{button ,AL(`CLS_CorelScript;FNC_SetSize')} **Related Topics**

CorelScript.SetReferencePoint

Function **SetReferencePoint**(ByVal **ReferencePoint** AS Long) AS Long

0 CorelScript

Description

This command sets the specified Reference Point for a selected object. The reference point is used to set the object handle for subsequent commands such as .SetPosition.

Parameter	Description
.**ReferencePoint**	Lets you specify the reference point to set. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetReferencePoint command.

Example

```
.CreateRectangle 1250000, -1000000, 750000, -500000, 0
.SetReferencePoint 9
.SetPosition 0, 0
```

The above example creates a rectangle, sets its reference point to the center and positions it in the center of the page.

{button ,AL(`CLS_CorelScript;FNC_SetReferencePoint')} **Related Topics**

CorelScript.SetApplyToDuplicate

Function **SetApplyToDuplicate**(ByVal **ApplyToDuplicate** AS Boolean) AS Long

0 CorelScript

Description

This command opens and closes a block of object-duplicating commands. An object must be selected to use this command. The duplicated object can be repositioned, resized, skewed, or rotated.

Parameter	Description
.**ApplyToDuplicate**	Set to TRUE (-1) to open a block of object-duplicating commands. Set to FALSE (0) to close the block.

Note

- The following commands can be used to duplicate objects within the .SetApplyToDuplicate block:
  .SetPosition
  .SkewObject
  .SetSize
  .RotateObject
  0 The duplicated object is selected.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.SetPosition 55555, 900000
.SetApplyToDuplicate TRUE
.SetPosition 0, 0 'Creates another object
.ApplyUniformFillColor 2, 255, 0, 0, 0
.SetPosition 55555, 100000 'Creates another object
.ApplyUniformFillColor 2, 0, 255, 0, 0
.SkewObject -15000000, 2000000, 3 'Creates another object
.SetSize 444444, 555555 'Creates another object
.RotateObject 45000000, 0, 0, 0 'Creates another object
.SetApplyToDuplicate FALSE
.SetPosition 0, 0
```

The above example creates an ellipse then creates 5 more ellipses in the SetApplyToDuplicate block.

{button ,AL(`CLS_CorelScript;FNC_SetApplyToDuplicate')} **Related Topics**

CorelScript.CopyToClipboard

Function **CopyToClipboard**() AS Long

0 CorelScript

Description

This command places a copy of the selected object(s) or text onto the Clipboard.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 2
.PasteFromClipboard
```

The above example copies a rectangle to the Clipboard, inserts 2 pages, then pastes the contents of the Clipboard to the third page.

{button ,AL(`CLS_CorelScript;FNC_CopyToClipboard')} **Related Topics**

CorelScript.PasteFromClipboard

Function **PasteFromClipboard**() AS Boolean

0 CorelScript

Description

This command places a copy of the object(s) on the Clipboard into your drawing.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 2
.PasteFromClipboard
```

The above example copies a rectangle to the Clipboard, inserts 2 pages, then pastes the contents of the Clipboard in to the last page inserted.

{button ,AL(`CLS_CorelScript;FNC_PasteFromClipboard')} **Related Topics**

CorelScript.ApplyStyle

Function **ApplyStyle**(ByVal **Style** AS String) AS Long

0 CorelScript

Description

This command lets you apply a style to the selected object.

Parameter	Description
.**Style**	Lets you specify the name of the style.

Example

```
.SelectAllObjects
.ApplyStyle "Default Graphic"
```

The above example applies the 'Default Graphic' style to all selected objects.

{button ,AL(`CLS_CorelScript;FNC_ApplyStyle')} **Related Topics**

CorelScript.DeleteStyle

Function **DeleteStyle**(ByVal **Style** AS String) AS Long

0 CorelScript

Description

This command deletes styles. When you delete a style, objects with that style revert to the default style for that object type. The object's appearance does not change when it reverts to the default style.

Parameter	Description
.Style	Lets you specify the name of the style to delete.

Example

```
.DeleteStyle "Style 1"
```

The above example deletes the style named "Style 1."

{button ,AL(`CLS_CorelScript;FNC_DeleteStyle')} **Related Topics**

CorelScript.LoadStyles

Function **LoadStyles**(ByVal **StyleSheet** AS String) AS Long

0 CorelScript

Description

This command loads the styles from a template into the active drawing.

Parameter	Description
.**StyleSheet**	Lets you specify the name of the template to use.

Example

```
.LoadStyles "C:\mine.cdt"
```

The above example loads the styles from the template file "MINE.CDT" into the active document.

{button ,AL(`CLS_CorelScript;FNC_LoadStyles')} **Related Topics**

CorelScript.SaveTemplate

Function **SaveTemplate**(ByVal **StyleSheet** AS String) AS Long

0 CorelScript

Description

This command lets you save the styles in the active document as a template.

Parameter	Description
.**StyleSheet**	Lets you specify the name of the Style Sheet to save.

Example

```
.SaveTemplate "C:\TMPLATE1.CDT"
```

The above example saves a template named "TMPLATE1.CDT".

{button ,AL(`CLS_CorelScript;FNC_SaveTemplate')} **Related Topics**

CorelScript.SetOutlineColor

Function **SetOutlineColor**() AS Long

0 CorelScript

Description

This command sets the color to be applied to the outline.

Example

```
.StoreColor DRAW_COLOR_CMYK100, 0, 0, 255, 0
.SetOutlineColor
```

The above example sets the outline color to yellow.

{button ,AL(`CLS_CorelScript;FNC_SetOutlineColor')} **Related Topics**

CorelScript.UnSelectAll

Function **UnSelectAll**() AS Long

0 CorelScript

Description

This command deselects all objects.

Example

```
.UnSelectAll
```

The above example deselects all selected object(s).

{button ,AL(`CLS_CorelScript;FNC_UnSelectAll')} **Related Topics**

CorelScript.CreateRectangle

Function **CreateRectangle**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **CornerRadius** AS Long, ByVal **CornerRadius2** AS Long, ByVal **CornerRadius3** AS Long, ByVal **CornerRadius4** AS Long) AS Long

0 CorelScript

Description

This command draws rectangles and squares.

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the rectangle in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the rectangle in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the rectangle in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the X-coordinate of the lower-right corner of the rectangle in tenths of a micron, relative to the center of the page.
3	.**CornerRadius** (optional) Lets you specify the radius used to create the rounded corners in tenths of a micron.
4	.**CornerRadius2** (optional) Lets you specify the radius used to create the rounded corners in tenths of a micron.
5	.**CornerRadius3** (optional) Lets you specify the radius used to create the rounded corners in tenths of a micron.
6	.**CornerRadius4** (optional) Lets you specify the radius used to create the rounded corners in tenths of a micron.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
```

The above example creates a rectangle.

```
FOR count% = 1 TO 8
.CreateRectangle 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000* count),
-500000+( 200000* count), 0
NEXT count
```

The above example creates 8 rectangles.

{button ,AL(`CLS_CorelScript;FNC_CreateRectangle')} **Related Topics**

CorelScript.CreateEllipse

Function **CreateEllipse**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **StartAngle** AS Long, ByVal **EndAngle** AS Long, ByVal **Arc** AS Boolean) AS Long

0 CorelScript

Description

This command draws ellipses and circles.

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the X-coordinate of the lower-right corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
3	.**StartAngle** If .CreateEllipse is used to create an arc, .StartAngle specifies the starting angle in degrees.
4	.**EndAngle** If .CreateEllipse is used to create an arc, .EndAngle specifies the end angle, in degrees.
5	.**Arc** Lets you specify whether to draw the ellipse as a pie or an arc. Set to TRUE (-1) to turn the ellipse into a pie. Set to FALSE (0) to draw the ellipse as an arc.

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
```

The above example creates an ellipse.

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000* count),
-500000+( 200000* count), 0, 0, 0
next count
```

The above example creates 4 ellipses.

{button ,AL(`CLS_CorelScript;FNC_CreateEllipse')} **Related Topics**

CorelScript.ApplyPreset

Function **ApplyPreset**(ByVal **PresetFileName** AS String, ByVal **PresetName** AS String) AS Long

0 CorelScript

Description

This command lets you load and apply a Preset.

Parameter	Description
.**PresetFileName**	Lets you specify the name of the Preset File.
0	.**PresetName** Lets you specify the name of the Preset.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyPreset "C:\CORELDRW.PST", "Button Blue"
```

The above example applies the specified preset fill to the rectangle.

{button ,AL(`CLS_CorelScript;FNC_ApplyPreset')} **Related Topics**

CorelScript.SelectNextObject

Function **SelectNextObject**(ByVal **SelectInsideGroup** AS Boolean) AS Boolean

0 CorelScript

Description

This command lets you select the next object in the drawing. Repeat this command until the object you want is selected.

Parameter	Description
.**SelectInsideGroup**	Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE (0) to disable this option.

Example

```
.SelectNextObject -1
```

The above example selects the next object in the drawing. If that object is in a group, it can be selected.

{button ,AL(`CLS_CorelScript;FNC_SelectNextObject')} **Related Topics**

CorelScript.SelectObjectAtPoint

Function **SelectObjectAtPoint**(ByVal **XPos** AS Long, ByVal **YPos** AS Long, ByVal **SelectInsideGroup** AS Boolean) AS Boolean

0 CorelScript

Description

This command toggles the selection of an object at the specified point. Using this command is the same as holding down SHIFT and clicking an object during a DRAW session.

Parameter	Description
.**XPos**	Lets you specify one of the X-coordinates of the selected object in tenths of a micron, relative to the center of the page.
0	.**YPos** Lets you specify one of the Y-coordinates of the selected object in tenths of a micron, relative to the center of the page.
1	.**SelectInsideGroup** Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE to disable this option.

Example

```
.CreateRectangle 1350000, -1000000, 1300000, 0, 0
.CreateRectangle 1000000, -750000, 500000, 100000, 0
.CreateRectangle 100000, -500000, -100000, 50000, 0
.CreateRectangle -750000, -500000, -250000, 50000, 0
.UnSelectAll
.SelectObjectAtPoint -750000, 500000, 0
.ApplyUniformFillColor 2, 255, 0, 0, 0
```

The above example creates four rectangles, then selects the second one and fills it with cyan.

{button ,AL(`CLS_CorelScript;FNC_SelectObjectAtPoint')} **Related Topics**

CorelScript.SelectObjectsInRect

Function **SelectObjectsInRect**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **IncludeIntersecting** AS Boolean) AS Boolean

0 CorelScript

Description

This command selects all objects found within the defined rectangular area

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the X-coordinate of the lower-right corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
3	.**IncludeIntersecting** Set to TRUE (-1) to included intersecting objects in the selection. Set to FALSE (0) to disable this option.

Example

```
.SelectObjectsInRect 1350000, -1000000, -1350000, 1000000, 0
```

The above example selects all objects within the specified rectangle.

{button ,AL(`CLS_CorelScript;FNC_SelectObjectsInRect')} **Related Topics**

CorelScript.SelectPreviousObject

Function **SelectPreviousObject**(ByVal **SelectInsideGroup** AS Boolean) AS Boolean

0 CorelScript

Description

This command lets you select the previously selected object in the drawing. Repeat this command until the object you want is selected. The objects are selected in the order in which they were created.

Parameter	Description
.**SelectInsideGroup**	Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE (0) to disable this option.

Example

```
.SelectPreviousObject -1
```

The above example selects the previous object in the group.

{button ,AL(`CLS_CorelScript;FNC_SelectPreviousObject')} **Related Topics**

CorelScript.FileSave

Function **FileSave**(ByVal **FileName** AS String, ByVal **ThumbnailSize** AS Long, ByVal **SaveSelectedOnly** AS Boolean, ByVal **FileVersion** AS Long, ByVal **IncludeCMXData** AS Boolean) AS Long

0 CorelScript

Description

This command saves the active document.

Parameter	Description
.**FileName**	Lets you specify the name of the file to save.
0	.**ThumbNailSize** Lets you specify the size of the thumbnail: 0 = None 1 = 1k (mono) 5 = 5k (color) 10 = 10k (color)
1	.**SaveSelectedOnly** Set to TRUE (-1) to save selected items only. Set to FALSE (0) to save entire document.
2	.**FileVersion** Lets you specify the file version of the document being saved. 0 = Version 9.0 1 = Version 5.0 2 = Version 6.0 3 = Version 7.0 4 = Version 8.0
3	.**IncludeCMXData** Set to TRUE (-1) to include CMX data with the saved file. Set to FALSE (0) to disable this feature.

{button ,AL(`CLS_CorelScript;FNC_FileSave')} **Related Topics**

CorelScript.OrderToFront

Function **OrderToFront**() AS Long

0 CorelScript

Description

This command rearranges the stacking order by moving the selected object to the front of the layer.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.SelectPreviousObject 0
.OrderToFront
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The rectangle is then selected and ordered to the front of the drawing.

{button ,AL(`CLS_CorelScript;FNC_OrderToFront')} **Related Topics**

CorelScript.OrderToBack

Function **OrderToBack**() AS Long

0 CorelScript

Description

This command rearranges the stacking order by moving the selected object to the back of the screen. Areas of the object overlapped by other objects with fills are "knocked out" so that they will not print.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.OrderToBack
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The ellipse, still selected, is ordered to the back of the drawing.

{button ,AL(`CLS_CorelScript;FNC_OrderToBack')} **Related Topics**

CorelScript.OrderForwardOne

Function **OrderForwardOne**() AS Long

0 CorelScript

Description

This command rearranges the stacking order by moving the selected object up one position.

Example

```
.SelectObjectOfCDRStaticID Six&
.OrderForwardOne
```

The above example orders the selected object forward one position.

{button ,AL(`CLS_CorelScript;FNC_OrderForwardOne')} **Related Topics**

CorelScript.OrderBackOne

Function **OrderBackOne**() AS Long

0 CorelScript

Description

This command rearranges the stacking order by moving the selected object back one position.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.OrderBackOne
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The ellipse, still selected, is ordered back one position in the drawing.

{button ,AL(`CLS_CorelScript;FNC_OrderBackOne')} **Related Topics**

CorelScript.OrderReverseOrder

Function **OrderReverseOrder**() AS Long

0 CorelScript

Description

This command reverses the stacking order of the selected object(s).

Example

```
.SelectAllObjects
.OrderReverseOrder
```

The above example reverses the order of all the objects.

{button ,AL(`CLS_CorelScript;FNC_OrderReverseOrder')} **Related Topics**

CorelScript.Group

Function **Group**() AS Long

0 CorelScript

Description

This command groups all selected objects together to allow them to be selected and manipulated as a single object.

Example

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000* count),
-500000+( 200000* count), 0, 0, 0
next count
.SelectAllObjects
.Group
.ApplyUniformFillColor 5, 0, 0, 255, 0
```

The above example groups the four ellipses together so that they are treated as one object, and applies a blue uniform fill to all four.

{button ,AL(`CLS_CorelScript;FNC_Group')} **Related Topics**

CorelScript.Ungroup

Function **Ungroup**() AS Long

0 CorelScript

Description

This command breaks up the selected group into its individual objects. If you have more than one sublevel of grouping, Ungroup breaks up one level of grouping at a time.

Example

```
.Ungroup
```

The above example breaks up the grouped object into its individual object components.

{button ,AL(`CLS_CorelScript;FNC_Ungroup')} **Related Topics**

CorelScript.Combine

Function **Combine**() AS Long

0 CorelScript

Description

This command combines the selected curve or line segments into a single object. If you use Combine on rectangles, ellipses, polygons, or text, CorelDRAW converts them to curves before converting them into a single curve object. However, when text is combined with other text it is not converted to curves; it is converted to larger blocks of text.

Example

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000* count),
-500000+( 200000* count), 0, 0, 0
next count
.SelectAllObjects
.Combine
.ApplyUniformFillColor 2, 0, 255, 0, 0
```

The above example creates 4 ellipses, then combines them before applying a fill.

{button ,AL(`CLS_CorelScript;FNC_Combine')} **Related Topics**

CorelScript.BreakApart

Function **BreakApart**() AS Long

0 CorelScript

Description

This command converts an object made up of multiple subpaths into individual curve objects.

Example

```
.BreakApart
```

The above example breaks apart the selected object into individual curve objects.

{button ,AL(`CLS_CorelScript;FNC_BreakApart')} **Related Topics**

CorelScript.Weld

Function **Weld**(ByVal **LeaveTarget** AS Boolean, ByVal **LeaveModifiers** AS Boolean) AS Long

0 CorelScript

Description

This command joins overlapping objects at points where their paths intersect. Although not necessarily apparent in editable preview, welding also removes sections of the path between those intersect points. The resulting curve object assumes the fill and outline attributes of the bottom object of the selected group of objects. If you marquee-select the objects, CorelDRAW will outline and fill the welded object with the attributes of the most recently created object.

Parameter	Description
.**LeaveTarget**	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
0	.**LeaveModifiers** Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Weld -1, -1
```

The above example welds the selected object group. Both the target object and the modifier object are left in the document.

{button ,AL(`CLS_CorelScript;FNC_Weld')} **Related Topics**

CorelScript.Intersection

Function **Intersection**(ByVal **LeaveTarget** AS Boolean, ByVal **LeaveModifiers** AS Boolean) AS Long

0 CorelScript

Description

This command creates a new object using the area common to two or more overlapping objects. Intersection joins their paths at the points where they intersect. The resulting curve object assumes the fill and outline attributes of the last selected object.

Parameter	Description
.**LeaveTarget**	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
0	.**LeaveModifiers** Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Intersection, -1, -1
```

The above example selects all objects and creates a new object(s) using the area common to overlapping objects. Both the target object and the modifier object are left in the document.

{button ,AL(`CLS_CorelScript;FNC_Intersection')} **Related Topics**

CorelScript.Trim

Function **Trim**(ByVal **LeaveTarget** AS Boolean, ByVal **LeaveModifiers** AS Boolean) AS Long

0 CorelScript

Description

This command lets you trim selected objects. Trimming two or more overlapping objects reshapes the last object selected. Trimming separates the paths at points where the objects overlap. Initially, the trimmed object may appear no different than it did before trimming. However, closer inspection will show that new nodes appear where the object was trimmed. Move the trimmed objects apart to see the full effect of the trim.

Parameter	Description
.**LeaveTarget**	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
0 .**LeaveModifiers**	Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Trim ,-1, -1
```

The above example trims the selected objects. Both the target object and the modifier object are left in the document.

{button ,AL(`CLS_CorelScript;FNC_Trim')} **Related Topics**

CorelScript.Separate

Function **Separate**() AS Long

0 CorelScript

Description

This command separates original objects from intermediate shapes.

Example

```
.Separate
```

The above example separates a combined object into its individual component object(s).

{button ,AL(`CLS_CorelScript;FNC_Separate')} **Related Topics**

CorelScript.ConvertToCurves

Function **ConvertToCurves**() AS Long

0 CorelScript

Description

This command converts the selected polygon, rectangle, ellipse, or text object to a series of curves you can shape with the Shape tool.

Example

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.ConvertToCurves
```

The above example converts the selected rectangle to a curve object.

{button ,AL(`CLS_CorelScript;FNC_ConvertToCurves')} **Related Topics**

CorelScript.DropSymbol

Function **DropSymbol**(ByVal **SymbolLibrary** AS String, ByVal **SymbolNumber** AS Long, ByVal **Tile** AS Boolean, ByVal **XPosOrGridSize** AS Long, ByVal **YPosOrGridSize** AS Long, ByVal **ProportionalSizing** AS Boolean, ByVal **SymbolSize** AS Long) AS Long

0 CorelScript

Description

This command positions the specified symbol at the defined position or the specified grid position.

Parameter	Description
.**SymbolLibrary**	Lets you specify the name of the Symbol Library. Refer to the Symbols dialog box for more details.
0	.**SymbolNumber** Lets you specify the Symbol Index Number, which identifies the selected symbol. Refer to the Symbols dialog box for more details.
1	.**Tile**Set to TRUE (-1) to create a pattern from the selected symbol that fills the page. Set to FALSE (0) to disable this option. Note that the tiled symbols are clones of the upper left symbol.
2	.**XPosOrGridSize** Lets you specify the X-coordinate or grid position at which to place the symbol, in tenths of a micron.
3	.**YPosOrGridSize** Lets you specify the Y-coordinate or grid position at which to place the symbol, in tenths of a micron.
4	.**ProportionalSizing** Set to TRUE (-1) to enable proportional sizing of the symbol. Set to FALSE (0) to disable this option.
5	.**SymbolSize** Lets you specify the size of the symbol in tenths of a micron. The symbol can be resized after it's been added to your drawing.

Example

```
.DropSymbol "Animals 1", 42, 0, 0, 0, 0, 1000000
```

The above example places a kangaroo symbol in the center of the page.

{button ,AL(`CLS_CorelScript;FNC_DropSymbol')} **Related Topics**

CorelScript.DuplicateObject

Function **DuplicateObject**(ByVal **XOffset** AS Long, ByVal **YOffset** AS Long) AS Long

0 CorelScript

Description

This command adds a copy of the selected object(s) to the current drawing. By default, the copy is placed on top of the original, offset up and to the right. It is also selected automatically.

Parameter	Description
.**XOffset**	Lets you specify the horizontal distance to offset the duplicate object.
0 .**YOffset**	Lets you specify the vertical distance to offset the duplicate object.

Example

```
.CreateRectangle 1082025, -333882, 272052, 500823, 0, 0, 0, 0
0   .StoreColor 5002, 100, 0, 100, 0, 0, 0, 0, 0
1   .ApplyUniformFillColor
2   .DuplicateObject
```

The above example creates a rectangle and fills it with a color, then duplicates the object.

Note

- The created object is created on top of the object you duplicated.

{button ,AL(`CLS_CorelScript;FNC_DuplicateObject')} **Related Topics**

CorelScript.CloneObject

Function **CloneObject**(ByVal **XOffset** AS Long, ByVal **YOffset** AS Long) AS Long

0 CorelScript

Description

This command copies the selected object and offsets the copy from the original. Most changes applied to the original object (called the "master") are automatically applied to the copy (called the "clone"). For example, if you change the master's fill, the clone's fill will change as well. If you change the attributes of the clone, the attribute you change will no longer depend on the master's attributes. For example, after you change a clone's fill, its fill will no longer change when you change the master's fill. Likewise, if you stretch a clone, it will no longer stretch when you stretch its master.

Parameter	Description
.**XOffset**	Lets you specify the horizontal distance to offset the clone object.
0 .**YOffset**	Lets you specify the vertical distance to offset the clone object.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CloneObject
```

The above example creates an ellipse, then makes a clone.

{button ,AL(`CLS_CorelScript;FNC_CloneObject')} **Related Topics**

CorelScript.DeleteObject

Function **DeleteObject**() AS Long

0 CorelScript

Description

This command deletes selected objects.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CreateRectangle 750000, -750000, 0, 0, 0
.DeleteObject
```

The above example deletes the selected object. Since the rectangle is the last object created, it is selected and gets deleted.

{button ,AL(`CLS_CorelScript;FNC_DeleteObject')} **Related Topics**

CorelScript.AlignObjects

Function **AlignObjects**(ByVal **HorizontalAlignment** AS Long, ByVal **VerticalAlignment** AS Long) AS Long

0 CorelScript

Description

This command aligns the selected objects according to the last selected object.

Parameter	Description
.**HorizontalAlignment**	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
0	.**VerticalAlignment** Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignObjects 2, 0
```

The above example aligns the selected objects according to the last selected object.

{button ,AL(`CLS_CorelScript;FNC_AlignObjects')} **Related Topics**

CorelScript.AlignToGrid

Function **AlignToGrid**(ByVal **HorizontalAlignment** AS Long, ByVal **VerticalAlignment** AS Long) AS Long

0 CorelScript

Description

This command aligns the selected objects to the gridpoint nearest to the edge of the selection.

Parameter	Description
.**HorizontalAlignment**	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
0	.**VerticalAlignment** Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignToGrid 1, 0
```

The above example horizontally aligns all objects to a gridpoint, nearest to the left edge of the selection.

{button ,AL(`CLS_CorelScript;FNC_AlignToGrid')} **Related Topics**

CorelScript.AlignToCenterOfPage

Function **AlignToCenterOfPage**(ByVal **HorizontalAlignment** AS Long, ByVal **VerticalAlignment** AS Long) AS Long

0 CorelScript

Description

This command aligns selected objects to the center of the page.

Parameter	Description
.**HorizontalAlignment**	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
0	.**VerticalAlignment** Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignToCenterOfPage 0, 3
```

The above example vertically aligns all objects to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_AlignToCenterOfPage')} **Related Topics**

CorelScript.DistributeObjects

Function **DistributeObjects**(ByVal **HorizontalDistribution** AS Long, ByVal **VerticalDistribution** AS Long, ByVal **ObjectOrPageExtents** AS Long) AS Long

0 CorelScript

Description

This command distributes selected objects.

Parameter	Description
.**HorizontalDistribution**	Lets you specify the type of horizontal distribution. 0 = None 1 = Right edges of object 2 = Left edges of object 3 = Center edges of object 4 = Space between objects
0	.**VerticalDistribution** Lets you specify the type of vertical distribution. 0 = None 1 = Top edges of object 2 = Bottom edges of object 3 = Center edges of object 4 = Space between objects
1	.**ObjectOrPageExtents** Lets you specify the type of distribution. 0 = Extent of Selection 1 = Extent of Page

Example

```
.SelectAllObjects
.DistributeObjects 3, 3, 1
```

The above example distributes the selected objects to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_DistributeObjects')} **Related Topics**

CorelScript.RotateObject

Function **RotateObject**(ByVal **Angle** AS Long, ByVal **UseObjectsCenter** AS Boolean, ByVal **XCenter** AS Long, ByVal **YCenter** AS Long) AS Long

0 CorelScript

Description

This command lets you rotate the selected object.

Parameter	Description
.**Angle**	Lets you specify the angle of rotation of the selected object, expressed in millionths of degrees. Negative values rotate the object clockwise from its current position; positive values rotate it counterclockwise. e.g., 45 degrees clockwise = -45000000
0	.**UseObjectsCenter** Set to TRUE (-1) to enable rotation around the center of the object. Set to FALSE (0) to disable this option.
1	.**XCenter** Lets you specify the logical X-coordinate of the center of the object to be rotated in tenths of a micron, relative to the center of the page.
2	.**YCenter** Lets you specify the logical Y-coordinate of the center of the object to be rotated in tenths of a micron, relative to the center of the page.

Note

- You can use the ANGLECONVERT function to specify angle measurements.

Example

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.RotateObject 45000000, -1, 0,0
```

The above example rotates the rectangle 45 degrees counter clockwise.

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.RotateObject -45000000, 0, -500000, 500000
```

The above example rotates the rectangle 45 degrees clockwise about the specified point.

{button ,AL(`CLS_CorelScript;FNC_RotateObject')} **Related Topics**

CorelScript.SkewObject

Function **SkewObject**(ByVal **XAngle** AS Long, ByVal **YAngle** AS Long, ByVal **Reference** AS Long) AS Long

0 CorelScript

Description

This command lets you skew the selected object.

Parameter	Description
.**XAngle**	Lets you specify the amount of horizontal skew (skew along the X-axis), in millionths of degrees. Positive angles result in counter-clockwise skew. Negative angles result in clockwise skew.
0	.**YAngle** Lets you specify the amount of vertical skew (skew along the Y-axis), in millionths of degrees. Positive angles result in counter-clockwise skew. Negative angles result in clockwise skew.
1	.**Reference** Lets you specify the reference point of the object to be skewed. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

Note

- You can use the ANGLECONVERT function to specify angle measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.SkewObject -15000000, 20000000, 3
```

The above example creates a rectangle, horizontally skews it 15 degrees clockwise and vertically skews it 20 degrees counterclockwise. The reference point for skewing is the upper-left position.

{button ,AL(`CLS_CorelScript;FNC_SkewObject')} **Related Topics**

CorelScript.InsertPages

Function **InsertPages**(ByVal **BeforePage** AS Boolean, ByVal **NumberOfPages** AS Long, ByVal **StartPageNumber** AS Long) AS Long

0 CorelScript

Description

This command inserts the specified number of pages into the current drawing.

Parameter	Description
.**BeforeCurrentPage**	Set to TRUE (-1) to position insertion point before the current page. Set to FALSE (0) to position insertion point after the current page.
0	.**NumberOfPages** Lets you specify the number of pages to insert.
1	.**StartPageNumber** (optional) Lets you specify the page number from where to insert pages.

Example

```
.InsertPages 0, 4, 5
```

The above example inserts 4 pages after the current page, starting on the fifth page.

{button ,AL(`CLS_CorelScript;FNC_InsertPages')} **Related Topics**

CorelScript.DeletePages

Function **DeletePages**(ByVal **UnusedParameter** AS Boolean, ByVal **NumberOfPages** AS Long, ByVal **StartPageNumber** AS Long) AS Long

0 CorelScript

Description

This command deletes pages from the current drawing.

Parameter	Description
.**UnusedParameter**	This parameter is not used
0	.**NumberOfPages** Lets you specify the number of pages to delete. Note: The current page is included in the deletion.
1	.StartPage Lets you specify the page number to begin deleting pages to delete.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 4
.PasteFromClipboard
.DeletePages , 2, 2
```

The above example inserts 4 pages after the current page, pastes the contents of the Clipboard on the fourth page, then deletes two pages starting on the second page.

{button ,AL(`CLS_CorelScript;FNC_DeletePages')} **Related Topics**

CorelScript.ExtractText

Function **ExtractText**(ByVal **DestinationFile** AS String) AS Boolean

0 CorelScript

Description

This command extracts the Artistic Text to a text file which can then be edited in any text editor and merged back into the document with MergeTextBack.

Parameter	Description
.**DestinationFile**	Lets you specify the name of the destination file.

Example

```
.CreateArtisticText "COREL DRAW"
.ExtractText "C:\TEXTFILE.TXT"
```

The above example extracts the text "COREL DRAW" to a text file named "TEXTFILE.TXT".

{button ,AL(`CLS_CorelScript;FNC_ExtractText')} **Related Topics**

CorelScript.MergeBackText

Function **MergeBackText**(ByVal **SourceFile** AS String) AS Boolean

0 CorelScript

Description

This command merges the extracted text back into the CorelDRAW document.

Parameter	Description
.**SourceFile**	Lets you specify the name of the source file to merge.

Example

```
.CreateArtisticText "COREL DRAW"
.ExtractText "C:\TEXTFILE.TXT"
.MergeBackText "C:\TEXTFILE.TXT"
```

The above example merges the extracted text from the file "TEXTFILE.TXT" back into the DRAW document.

{button ,AL(`CLS_CorelScript;FNC_MergeBackText')} **Related Topics**

CorelScript.GetObjectsCDRStaticID

Function **GetObjectsCDRStaticID**() AS Long

0 CorelScript

Description

This function returns the CDRStaticID of the selected object. If more than one object is selected, the function returns the CDRStaticID of the last selected object.

Return Value

Returns the following value:

- the CDRStaticID of the selected object

Note

- Every object you create has a unique CDRStaticID in a document.

Example

```
.CreateRectangle 750000, -600000, 250000, -100000, 0
IDRect& = .GetObjectsCDRStaticID()
.SelectObjectOfCDRStaticID IDRect&
```

The above example demonstrates object selection using the object's CDRStaticID.

{button ,AL(`CLS_CorelScript;FNC_GetObjectsCDRStaticID')} **Related Topics**

CorelScript.SelectObjectOfCDRStaticID

Function **SelectObjectOfCDRStaticID**(ByVal **CorelDRAWID** AS Long) AS Boolean

0 CorelScript

Description

This command selects the object with the specified CDRStaticID.

Parameter	Description
.**CDRStaticID**	Lets you specify the CDRStaticID number of the object to select.

Example

```
.CreateRectangle 750000, -600000, 250000, -100000, 0
IDRect& = .GetObjectsCDRStaticID()
.SelectObjectOfCDRStaticID IDRect&
```

The above example demonstrates object selection using the object's CDRStaticID.

{button ,AL(`CLS_CorelScript;FNC_SelectObjectOfCDRStaticID')} **Related Topics**

CorelScript.GetUserDataField

Function **GetUserDataField**(ByVal **FieldName** AS String) AS String

0 CorelScript

Description

This function returns a specified user-data field of a selected object. If more than one object is selected, the function returns the specified user-data field of the last selected object.

Parameter	Description
ReturnString$	Returns the user data field of the selected object.
0	.**FieldName** Lets you specify the name of an object's user data field.

Example

```
u_d_f$="CDRStaticID"
data_field1$=.GetUserDataField (u_d_f)
data_field2$=.GetUserDataField ("Name")
```

The above example returns the value for the CDRStaticID and Name field of a selected object.

{button ,AL(`CLS_CorelScript;FNC_GetUserDataField')} **Related Topics**

CorelScript.SetUserDataField

Function **SetUserDataField**(ByVal **FieldName** AS String, ByVal **FieldValue** AS String) AS Long

0 CorelScript

Description

This command lets you set user data field values for selected objects.

Parameter	Description
.**FieldName**	Lets you specify the name of the user data field to set.
0	.**FieldValue** Lets you specify the value of the user data field to set.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
.SetUserDataField "Name", "MyObject"
```

The above example creates a rectangle and while it is still selected, sets its object name to "MyObject". Other common data fields for objects include cost and comments.

{button ,AL(`CLS_CorelScript;FNC_SetUserDataField')} **Related Topics**

CorelScript.FileClose

Function **FileClose**(ByVal **PromptUser** AS Boolean) AS Boolean

0 CorelScript

Description

This command closes the current drawing.

Return Value

Returns one of the following values:

- TRUE (-1) the file was closed
- FALSE (0) the file was not closed

Parameter	Description
.**PromptUser**	Set to TRUE (-1) to prompt the user before closing the file. Set to FALSE (0) to close the file without prompting the user.

Note

- This command must be preceded by the .FileSave command or changes will be lost.

Example

```
.FileClose TRUE
```

The above example prompts the user before closing the active CorelDRAW document.

```
.FileClose
```

The above example closes the active CorelDRAW document without prompting the user.

{button ,AL(`CLS_CorelScript;FNC_FileClose')} **Related Topics**

CorelScript.CreateArtisticText

Function **CreateArtisticText**(ByVal **NewText** AS String, ByVal **Left** AS Long, ByVal **Top** AS Long) AS Long

0 CorelScript

Description

This command allows you to create a text string with default text settings as Artistic Text. The left-most character of the Artistic Text is placed on the center of the page. Text created with the CreateArtisticText command can be modified using the SetArtisticText command.

Parameter	Description
.**NewText**	Lets you specify the name of the new text to create.
0	.**Left** Sets the left edge of the artistic text's position in tenths of a micron, relative to the center of the page.
1	.**Top** Sets the top edge of the artistic text's position in tenths of a micron, relative to the center of the page.

Example

```
.CreateArtisticText "CorelDRAW", 1000, 1000
```

The above example displays the text string "CorelDRAW" 100 microns to the left of the center of the page, and 100 microns above the center of the page.

{button ,AL(`CLS_CorelScript;FNC_CreateArtisticText')} **Related Topics**

CorelScript.SetArtisticText

Function **SetArtisticText**(ByVal **String** AS String) AS Long

0 CorelScript

Description

This command allows you to change selected Artistic Text text strings.

Parameter	Description
.**NewText**	Lets you specify the name of the new text to set.

Example

```
.FileNew
.CreateArtisticText "1"
.FilePrint
FOR i%=2 TO 10 STEP 1
.SetArtisticText i%
.FilePrint
NEXT i%
```

The above example creates the string "1" as Artistic text and then prints the document. Within the FOR...NEXT loop, the Artistic text is changed from the numbers 2 to 10. After each change in the Artistic text, the document is printed.

{button ,AL(`CLS_CorelScript;FNC_SetArtisticText')} **Related Topics**

CorelScript.ApplyNoFill

Function **ApplyNoFill**() AS Long

0 CorelScript

Description

This command removes the fill from the selected object, allowing objects behind it to show through.

Example

```
.SelectAllObjects
.ApplyNoFill
```

The above example removes the fill from all objects.

{button ,AL(`CLS_CorelScript;FNC_ApplyNoFill')} **Related Topics**

CorelScript.GetFillType

Function **GetFillType**() AS Long

0 CorelScript

Description

This function returns the Fill Type of a selected object. If more than one object is selected, the function returns the Fill Type of the last selected object.

Return Value

Returns one of the following values:

- 0 None
- 1 Uniform
- 2 Fountain
- 6 PostScript
- 7 Two-color
- 9 ColorBitmap
- 10 Vector
- 11 Texture

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .GetFillType command.

Example

```
.SelectObjectOfCDRStaticID IDRect&
fillType& = .GetFillType()
Message fillType&
```

The above example displays a number corresponding to the fill type of the selected object in a message box.

{button ,AL(`CLS_CorelScript;FNC_GetFillType')} **Related Topics**

CorelScript.GetObjectType

Function **GetObjectType**() AS Long

0 CorelScript

Description

This function returns a value that indicates the type of selected object. If more than one object is selected, the function returns the type of the last selected object.

Return Value

Returns one of the following values:

- 0 Reserved for future use
- 1 Rectangle
- 2 Ellipse
- 3 Curve
- 4 Text
- 5 Bitmap
- 6 Paragraph Text
- 7 OLE
- 9 Symmetrical Polygon
- 12 Grouped objects

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .GetObjectType command.

Example

```
objType& = .GetObjectType()
MESSAGE objType&
```

The above example displays a number that corresponds to the type of selected object in a message box.

{button ,AL(`CLS_CorelScript;FNC_GetObjectType')} **Related Topics**

CorelScript.BeginDrawCurve

Function **BeginDrawCurve**(ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This command sets the coordinates of the starting node when drawing curves in Freehand mode.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the starting node of the curve in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the starting node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .BeginDrawCurve command must be followed by a contiguous block of one or more DrawCurve commands, and one .EndDrawCurve command. The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`CLS_CorelScript;FNC_BeginDrawCurve')} **Related Topics**

CorelScript.DrawCurveMoveTo

Function **DrawCurveMoveTo**(ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This command sets the coordinates when drawing non-continuous curves in Freehand mode.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the point to move to without drawing in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the point to move to without drawing in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveMoveTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.DrawCurveMoveTo -500000, -500000
.DrawCurveLineTo 500000, 1000000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`CLS_CorelScript;FNC_DrawCurveMoveTo')} **Related Topics**

CorelScript.DrawCurveLineTo

Function **DrawCurveLineTo**(ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This command sets the coordinates when drawing continuous curves in Freehand mode.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the next node of the curve in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the next node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveLineTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`CLS_CorelScript;FNC_DrawCurveLineTo')} **Related Topics**

CorelScript.DrawCurveCurveTo

Function **DrawCurveCurveTo**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **XEnd** AS Long, ByVal **YEnd** AS Long) AS Long

0 CorelScript

Description

This command sets a node in a curve drawn in Freehand mode.

Parameter	Description
.**X1**	Lets you specify the X-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that was created in a preceding BeginDrawCurve or DrawCurveCurveTo command.
0	.**Y1** Lets you specify the Y-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that was created in a preceding BeginDrawCurve or DrawCurveCurveTo command.
1	.**X2** Lets you specify the X-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that is specified in the current DrawCurveCurveTo command.
2	.**Y2** Lets you specify the Y-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that is specified in the current DrawCurveCurveTo command.
3	.**XEnd** Lets you specify the X-coordinate of a node of the curve in tenths of a micron, relative to the center of the page.
4	.**YEnd** Lets you specify the X-coordinate of a node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveCurveTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveCurveTo 500000, 500000, 1000000 ,-500000, -500000, -500000
.DrawCurveCurveTo 600000, 600000, 1100000 ,-600000, -600000, -600000
.EndDrawCurve
```

The above example draws a curve.

{button ,AL(`CLS_CorelScript;FNC_DrawCurveCurveTo')} **Related Topics**

CorelScript.DrawCurveClosePath

Function **DrawCurveClosePath**() AS Long

0 CorelScript

Description

This command closes the path on the last node when drawing curves in Freehand mode.

Note

- The .DrawCurveClosePath command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveCurveTo 500000, 500000, 1000000 ,-500000, -500000, -500000
.DrawCurveClosePath
.EndDrawCurve
```

The above example draws an object in the shape of an uppercase "D".

{button ,AL(`CLS_CorelScript;FNC_DrawCurveClosePath')} **Related Topics**

CorelScript.EndDrawCurve

Function **EndDrawCurve**() AS Long

0 CorelScript

Description

This command ends a set of curve creation commands that began with the .BeginDrawCurve command.

Note

- The DrawCurve commands include:
  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`CLS_CorelScript;FNC_EndDrawCurve')} **Related Topics**

CorelScript.NewLayer

Function **NewLayer**(ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command lets you create a new layer and assign a name.

Parameter	Description
.**LayerName**	Lets you specify the name of the new layer.

Example

```
.NewLayer "NewLayer1"
```

The above example creates a new layer named "NewLayer1."

{button ,AL(`CLS_CorelScript;FNC_NewLayer')} **Related Topics**

CorelScript.DeleteLayer

Function **DeleteLayer**(ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command deletes the active layer and any objects on it.

Parameter	Description
.**LayerName**	Lets you specify the name of the layer to delete. This parameter is optional. If no layer name is specified the active layer is deleted.

Example

```
.MoveToLayer "NewLayer1"
.DeleteLayer
```

The above example moves to the layer named "NewLayer 1" and deletes it.

{button ,AL(`CLS_CorelScript;FNC_DeleteLayer')} **Related Topics**

CorelScript.SelectLayer

Function **SelectLayer**(ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command lets you select a layer, making it the active layer.

Parameter	Description
.**LayerName**	Lets you specify the name of the selected layer.

Example

```
.SelectLayer "NewLayer1"
```

The above example selects the layer named "NewLayer1" and makes it the active layer.

{button ,AL(`CLS_CorelScript;FNC_SelectLayer')} **Related Topics**

CorelScript.ChangeLayerName

Function **ChangeLayerName**(ByVal **NewLayerName** AS String) AS Long

0 CorelScript

Description

This command lets you assign a new name to the active layer.

Parameter	Description
.**NewLayerName**	Lets you specify the new name of the Layer.

Example

```
.ChangeLayerName "NewName"
```

The above example changes the layer name to "NewName."

{button ,AL(`CLS_CorelScript;FNC_ChangeLayerName')} **Related Topics**

CorelScript.SetOptionsForAllPages

Function **SetOptionsForAllPages**(ByVal **AllPages** AS Boolean) AS Long

0 CorelScript

Description

This command enables CorelDRAW options to be set for all pages.

Parameter	Description
.**AllPages**	Set to TRUE (-1) to enable options to be set for all pages. Set to FALSE (0) to disable this option.

{button ,AL(`CLS_CorelScript;FNC_SetOptionsForAllPages')} **Related Topics**

CorelScript.SetMultiLayer

Function **SetMultiLayer**(ByVal **MultiLayer** AS Boolean) AS Long

0 CorelScript

Description

This command lets you select objects on all layers that are not locked or invisible.

Parameter	Description
.**MultiLayer**	Set to TRUE (-1) to enable selection of objects across all layers except those which are locked or invisible. Set to FALSE (0) to disable selection of objects across all layers—only objects on the current layer can be selected.

{button ,AL(`CLS_CorelScript;FNC_SetMultiLayer')} **Related Topics**

CorelScript.SetToMasterLayer

Function **SetToMasterLayer**(ByVal **Master** AS Boolean, ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command lets you set the selected object to a master layer. When you want the same element, for example, a company logo, to appear on every page of a document, use this command to set the "master layers" to contain the repeating elements.

Parameter	Description
.**Master**	Set to TRUE (-1) to enable, applying the Master Layer template to all layers. Set to FALSE (0) to disable this option.
0	.**LayerName** (optional) Lets you specify the layer name.

Example

```
.CreateRectangle 1350000, -1000000, 750000, -500000, 0
.SetToMasterLayer -1
```

The above example sets the rectangle to the master layer.

{button ,AL(`CLS_CorelScript;FNC_SetToMasterLayer')} **Related Topics**

CorelScript.SetLayerVisible

Function **SetLayerVisible**(ByVal **Visible** AS Boolean, ByVal **LayerName** AS String, ByVal **PageNum** AS Long) AS Long

0 CorelScript

Description

This command makes objects on a layer visible or invisible.

Parameter	Description
.**Visible**	Set to TRUE (-1) to make the current layer visible. Set to FALSE (0) to make the current layer invisible.
0	.**LayerName** (optional) Lets you specify the layer name.
1	.**PageNum** (optional) Specifes the page number. 0=Master

Note

- If .SetOptionsForAllPages is set TRUE (-1), then the .SetLayerVisible command applies to all pages.

Example

```
.SetLayerVisible -1, 3, 2
```

The above example makes the third layer on the second page visible.

{button ,AL(`CLS_CorelScript;FNC_SetLayerVisible')} **Related Topics**

CorelScript.SetLayerPrintable

Function **SetLayerPrintable**(ByVal **Printable** AS Boolean, ByVal **LayerName** AS String, ByVal **PageNum** AS Long) AS Long

0 CorelScript

Description

This command enables or disables printing of objects on the current layer.

Parameter	Description
.**Printable**	Set to TRUE (-1) to enable printing of the current layer. Set to FALSE (0) to disable printing of the current layer.
0	.**LayerName** (optional) Lets you specify the layer name.
1	.**PageNum** (optional) Specifes the page number. 0=Master

Note

- If .SetOptionsForAllPages is set TRUE (-1), then the .SetLayerPrintable command applies to all pages.

Example

```
.SetLayerPrintable 0, 3, 2
```

The above example disables printing of the third layer on the second page.

{button ,AL(`CLS_CorelScript;FNC_SetLayerPrintable')} **Related Topics**

CorelScript.SetLayerLocked

Function **SetLayerLocked**(ByVal **Locked** AS Boolean, ByVal **LayerName** AS String, ByVal **PageNum** AS Long) AS Long

0 CorelScript

Description

This command enables or disables selection of objects on a layer. Locking a layer prevents objects on it from being accidentally moved or changed in any way. You cannot add new objects to a locked layer.

Parameter	Description	
.**Locked**	Set to TRUE (-1) to lock a layer, preventing objects on it from being accidentally moved or changed in any way. You cannot add new objects to a locked layer. Set to FALSE (0) to unlock a layer.	
0	.**LayerName** (optional)	Lets you specify the layer name.
1	.**PageNum** (optional)	Specifes the page number. 0=Master

Example

```
.SetLayerLocked -1, 3, 2
```

The above example locks third layer on the second page.

{button ,AL(`CLS_CorelScript;FNC_SetLayerLocked')} **Related Topics**

CorelScript.SetColorOverride

Function **SetColorOverride**(ByVal **Override** AS Boolean) AS Long

0 CorelScript

Description

This command outlines objects on a layer in the selected color. Objects on the selected layer will appear with a wireframe outline of the chosen color.

Parameter	Description
.**Override**	Set to TRUE (-1) to outline objects on a layer in the selected color. Set to FALSE (0) to disable this option.

Example

```
.StoreColor DRAW_COLORMODEL_PANTONE, 3, 255, 0, 0
0   .SetColorOverride
```

The above example sets the override color to cyan.

{button ,AL(`CLS_CorelScript;FNC_SetColorOverride')} **Related Topics**

CorelScript.MoveToLayer

Function **MoveToLayer**(ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command moves the selected object to the layer selected in the Layers list.

Parameter	Description
.**LayerName**	Lets you specify the name of the destination layer.

Example

```
.MoveToLayer "NewLayer1"
```

The above example moves the selected object(s) to the layer named "NewLayer1."

{button ,AL(`CLS_CorelScript;FNC_MoveToLayer')} **Related Topics**

CorelScript.CopyToLayer

Function **CopyToLayer**(ByVal **LayerName** AS String) AS Long

0 CorelScript

Description

This command places a copy of the selected object on the layer indicated in the LayerName.

Parameter	Description
.**LayerName**	Lets you specify the name of the destination layer.

Example

```
.CreateRectangle -200000, 200000, -900000, 900000, 0
.CopyToLayer "Layer2"
```

The above example creates a rectangle and copies it to "Layer2."

{button ,AL(`CLS_CorelScript;FNC_CopyToLayer')} **Related Topics**

CorelScript.SetPageSize

Function **SetPageSize**(ByVal **Width** AS Long, ByVal **Height** AS Long) AS Long

0 CorelScript

Description

This command lets you set the page size for the document.

Parameter	Description
.**Width**	Lets you specify the new page width in tenths of a micron.
0 .**Height**	Lets you specify the new page height in tenths of a micron.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.SetPageSize 1000000,1350000
```

The above example sets the page size to 1,000,000 microns wide by 1,350,000 microns high (or 3.94 inches by 5.31 inches).

{button ,AL(`CLS_CorelScript;FNC_SetPageSize')} **Related Topics**

CorelScript.SetPageSizeFromPrinter

Function **SetPageSizeFromPrinter**() AS Long

0 CorelScript

Description

This command sets the page size and orientation of the current document to the current settings of the default printer.

Example

```
.SetPageSizeFromPrinter
```

The above example queries the printer to set the page size.

{button ,AL(`CLS_CorelScript;FNC_SetPageSizeFromPrinter')} **Related Topics**

CorelScript.SetPageLayout

Function **SetPageLayout**(ByVal **LayoutType** AS Long) AS Long

0 CorelScript

Description

This command lets you specify a page layout.

Parameter	Description
.**LayoutType**	Lets you specify the style of the page layout: 1 = Full Page: Prints one full page per sheet. 2 = Book: Prints two pages per sheet, which you would cut down the middle. 3 = Booklet: Prints two pages per sheet, which you would fold vertically to obtain a side fold. 4 = Tent Card: Prints two pages per sheet, which you would fold horizontally to obtain a top fold. 5 = Side-Fold Card: Prints four pages per sheet, which you would fold first horizontally to create the top fold, then vertically to create the side fold. 6 = Top-Fold Card: Prints four pages per sheet, which you would fold first vertically to create the side fold, then horizontally to create the top fold.

Example

```
.SetPageLayout 3
```

The above example sets the page layout to booklet style.

{button ,AL(`CLS_CorelScript;FNC_SetPageLayout')} **Related Topics**

CorelScript.DisplayFacingPages

Function **DisplayFacingPages**(ByVal **FacingPages** AS Boolean, ByVal **LeftFirst** AS Boolean) AS Long

0 CorelScript

Description

This command displays two consecutive pages on the screen at the same time.

Parameter	Description
.**FacingPages**	Set to TRUE (-1) to display two consecutive pages on the screen at the same time. Working in this view allows you to draw objects that lie partially on both pages at the same time. Set to FALSE (0) to disable this option.
0 .**LeftFirst**	Set to TRUE (-1) to display odd pages on the left. Set to FALSE (0) to display odd pages on the right.

Example

```
.FileNew
.DisplayFacingPages 0, -1 'Displays one page
```

The above example displays one page.

```
.FileNew
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 4
.PasteFromClipboard
.DisplayFacingPages -1, -1 'Displays two pages
```

The above example displays facing pages with the current page on the left.

{button ,AL(`CLS_CorelScript;FNC_DisplayFacingPages')} **Related Topics**

CorelScript.SetPaperColor

Function **SetPaperColor**() AS Long

0 CorelScript

Description

This command lets you color the Preview screen (and the Drawing Window, if you are working in the Editable Preview) to approximate the paper you plan to print it on.

Example

```
.StoreColor DRAW_COLORMODEL_CMYK, 0, 255, 0, 0
0   .SetPaperColor
```

The above example sets the paper color to magenta.

{button ,AL(`CLS_CorelScript;FNC_SetPaperColor')} **Related Topics**

CorelScript.ShowPageBorder

Function **ShowPageBorder**(ByVal **ShowBorder** AS Boolean) AS Long

0 CorelScript

Description

This command enables and disables the page border.

Parameter	Description
.**ShowBorder**	Set to TRUE (-1) to show the page border. Set to FALSE (0) to suppress the page border.

Example

```
.ShowPageBorder -1
```

The above example shows the page border.

```
.ShowPageBorder 0
```

The above example hides the page border.

{button ,AL(`CLS_CorelScript;FNC_ShowPageBorder')} **Related Topics**

CorelScript.AddPageFrame

Function **AddPageFrame**() AS Long

0 CorelScript

Description

This command puts a printable background frame around the page.

Example

```
.AddPageFrame
```

The above example creates a frame around the new page.

{button ,AL(`CLS_CorelScript;FNC_AddPageFrame')} **Related Topics**

CorelScript.ApplyUniformFillColor

Function **ApplyUniformFillColor**() AS Long

0 CorelScript

Description

This command lets you apply a Uniform Fill Color to a selected object.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.StoreColor DRAW_COLORMODEL_CMYK, 100, 100, 0, 0, 0
.ApplyUniformFillColor
```

The above example creates an ellipse and uniformly fills it with cyan.

{button ,AL(`CLS_CorelScript;FNC_ApplyUniformFillColor')} **Related Topics**

CorelScript.GetUniformFillColor

Function **GetUniformFillColor**(ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef **V2** AS Long, ByRef **V3** AS Long, ByRef **V4** AS Long, ByRef **V5** AS Long, ByRef **V6** AS Long, ByRef **Density** AS Long) AS Long

0 CorelScript

Description

This function returns the Uniform Fill color attributes of a selected object. If more than one object is selected, the function returns the Uniform Fill color of the last selected object.

Parameters	Description
.**ColorModel**	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome 0 To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 1 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
V1	Description of 'V1' goes here
0	**V2** Description of 'V2' goes here
1	**V3** Description of 'V3' goes here
2	**V4** Description of 'V4' goes here
3	**V5** Description of 'V5' goes here
4	**V6** Description of 'V6' goes here
5	**Density** Description of 'Density' goes here

0 Example

1 Example of usage goes here

{button ,AL(`CLS_CorelScript;FNC_GetUniformFillColor')} **Related Topics**

CorelScript.ApplyFountainFill

Function **ApplyFountainFill**(ByVal **Type** AS Long, ByVal **CenterX** AS Long, ByVal **CenterY** AS Long, ByVal **Angle** AS Long, ByVal **Steps** AS Long, ByVal **Padding** AS Long, ByVal **Blend** AS Long, ByVal **Rate** AS Long) AS Long

0 CorelScript

Description

This command lets you apply a Fountain Fill to the selected object. If the Blend was Custom, then all intermediate colors will be lost unless the Blend applied is again Custom. If the existing fill is not fountain, the start color will be CMYK Black and the end color CMYK white.

Parameter	Description
.**Type**	Lets you specify the type of Fountain Fill to apply: 0 = Linear (default) 1 = Radial 2 = Conical 3 = Square
0	.**CenterX** Lets you specify the horizontal offset of the center of the fill. Valid values range from -100 to 100 percent. A value of -50% will place the center on the left edge of your object; a value of 50% will place it on the right edge.
1	.**CenterY** Lets you specify the vertical offset of the center of the fill. Valid values range from -100 to 100 percent. A value of -50% will place the center on the bottom edge of your object; a value of 50% will place it on the top edge.
2	.**Angle** Lets you specify the angle at which the fill is applied in tenths of degrees. Positive values will rotate the fill counter-clockwise, negative values will rotate it clockwise.
3	.**Steps** Lets you specify the number of steps you want. Lower values produce coarser fountains on screen which take less time to redraw.
4	.**Padding** Lets you specify the amount of padding to apply to the fill. Ignored for type 2. Valid values range from 0 to 45 percent.
5	.**Blend** Lets you specify the type of blending to apply to the fill. 0 = Direct (default) 1 = Rainbow CW 2 = Rainbow CCW 3 = Custom
6	.**Rate** Lets you specify the mid-point used to apply the fill. Valid values range from 1 to 99.

Note

- The Horizontal and Vertical Offset options are not available for linear fountain fills; set parameters to 0.
- The Angle option is not available for circular fountain fills; set parameter to 0.
- You can use the ANGLECONVERT function to specify angle measurements
- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyFountainFill command.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.ApplyFountainFill 1, -50, -50, 900, 20, 20, 2, 1
```

{button ,AL(`CLS_CorelScript;FNC_ApplyFountainFill')} **Related Topics**

CorelScript.RemoveFountainFillColor

Function **RemoveFountainFillColor**(ByVal **Position** AS Long) AS Long

0 CorelScript

Description

This command removes the currently selected Fountain Fill Color.

Parameter	Description
.**Position**	Lets you specify the position of the color to be removed. 0 and 100 are invalid values. For any other value, the color at that position is removed, if one exists. Existing fill must be a Fountain and Blend must be custom.

Example

```
.ApplyFountainFill 2, -50, -50, 900, 20, 20, 2, 0
.RemoveFountainFillColor 75
```

{button ,AL(`CLS_CorelScript;FNC_RemoveFountainFillColor')} **Related Topics**

CorelScript.GetFountainFill

Function **GetFountainFill**(ByRef **Type** AS Long, ByRef **CenterX** AS Long, ByRef **CenterY** AS Long, ByRef **Angle** AS Long, ByRef **Steps** AS Long, ByRef **Padding** AS Long, ByRef **Blend** AS Long, ByRef **Rate** AS Long, ByRef **NumColors** AS Long) AS Long

0 CorelScript

Description

This command returns the Fountain Fill attributes of a selected object. If more than one object is selected, the function returns the Fountain Fill attributes of the last selected object.

Parameter	Description
.**Type**	Returns the type of Fountain Fill: 0 = Linear (default) 1 = Radial 2 = Conical 3 = Square
0	.**CenterX** Returns the Horizontal Offset of the center of the fill. Valid values range from -100 to +100 percent. A value of -50% will place the center on the left edge of your object; a value of 50% will place it on the right edge.
1	.**CenterY** Returns the Horizontal Offset of the center of the fill. Valid values range from -100 to +100 percent. A value of -50% will place the center on the bottom edge of your object; a value of 50% will place it on the top edge.
2	.**Angle** Returns the angle at which the fill is applied in degrees. Positive values will rotate the fill counter-clockwise, negative values will rotate it clockwise.
3	.**Steps** Returns the number of stripes you want. Lower values produce coarser fountains on screen which take less time to redraw. Valid values range from 2 to 256.
4	.**Padding** Returns the amount of padding to apply to the fill. Ignored for type 2. Valid values range from 0 to 45 percent.
5	.**Blend** Returns the type of blending to apply to the fill. 0 = Direct (default) 1 = Rainbow CW 2 = Rainbow CCW 3 = Custom
6	.Rate Returns the rate method used to apply the fill.
7	.NumColors Returns the number of colors.

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.GetFountainFill fillType&, CX&, CY&, Angle&, Steps&, Pad&, Blend&, Rate&, Num&
MESSAGE fillType&
```

The above example returns Fountain Fill attributes and displays a number corresponding to the fill type in a message box.

{button ,AL(`CLS_CorelScript;FNC_GetFountainFill')} **Related Topics**

CorelScript.ApplyTwoColorFill

Function **ApplyTwoColorFill**(ByVal **FileName** AS String, ByVal **TileWidth** AS Long, ByVal **TileHeight** AS Long, ByVal **FirstTileOffsetX** AS Long, ByVal **FirstTileOffsetY** AS Long, ByVal **RowOffset** AS Boolean, ByVal **RowColumnOffset** AS Long, ByVal **SeamlessTiling** AS Boolean, ByVal **ScaleWithObject** AS Boolean, ByVal **RotationAngle** AS Long, ByVal **SkewAngle** AS Long) AS Long

0 CorelScript

Description

This command lets you apply a Two-Color fill to the selected object.

Parameter	Description
.**FileName**	Lets you specify the name of the two color fill file to use. See the Two-Color Bitmap Pattern dialog box for a list of valid file formats.
0	.**TileWidth** If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
1	.**TileHeight** If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
2	.**FirstTileOffsetX** X offset from center of object in the same units used by width & height.
3	.**FirstTileOffsetY** Y offset from center of object in the same units used by widht & height.
4	.**RowOffset** Set to TRUE (-1) to enable row offset. Set to FALSE (0) to enable column offset.
5	.**RowColumnOffset** Lets you specify the amount of row or column offsets. Valid values range from 0 to 100.
6	.**SeamlessTiling** Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
7	.**ScaleWithObject** Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
8	**.Rotation Angle** Lets you specify the amount in which the tile is rotated in millionths of degrees.
9	.**SkewAngle** Lets you specify the amount in which the tile is skewed in millionths of degrees.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyTwoColorFill "mybitmap.bmp", 5, 255, 0, 0, 0, 5, 0, 0, 0, 0, 500000, 500000, 100, 100, 0,
100, 0, 0, 0
```

The above example applies a two-color bitmap fill from the MYBITMAP.BMP file to the rectangle.

{button ,AL(`CLS_CorelScript;FNC_ApplyTwoColorFill')} **Related Topics**

CorelScript.ApplyFullColorFill

Function **ApplyFullColorFill**(ByVal **FileName** AS String, ByVal **TileWidth** AS Long, ByVal **TileHeight** AS Long, ByVal **FirstTileOffsetX** AS Long, ByVal **FirstTileOffsetY** AS Long, ByVal **RowOffset** AS Boolean, ByVal **RowColumnOffset** AS Long, ByVal **SeamlessTiling** AS Boolean, ByVal **ScaleWithObject** AS Boolean, ByVal **VectorBBoxTop** AS Long, ByVal **VectorBBoxBottom** AS Long, ByVal **VectorBBoxLeft** AS Long, ByVal **VectorBBoxRight** AS Long, ByVal **RotationAngle** AS Long, ByVal **SkewAngle** AS Long) AS Long

0 CorelScript

Description

This command lets you apply a Full-Color fill to the selected object.

Parameter	Description
.**FileName**	Lets you specify the name of the Fill file.
0	.**TileWidth** If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
1	.**TileHeight** If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
2	.**FirstTileOffsetX** X offset from center of object in the same units used by width & height.
3	.**FirstTileOffsetY** Y offset from center of object in the same units used by widht & height.
4	.**RowOffset** Set to TRUE (-1) to enable row offset. Set to FALSE (0) to enable column offset.
5	.**RowColumnOffset** Lets you specify the amount of row or column offsets. Valid values range from 0 to 100.
6	.**SeamlessTiling** Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
7	.**ScaleWithObject** Set to TRUE (-1) to scale the pattern with the object. Set to FALSE (0) to disable this option.
8	.**VectorBBoxTop** (optional) Lets you specify the top coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
9	.**VectorBBoxBottom** (optional) Lets you specify the bottom coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
10	.**VectroBBoxLeft** (optional) Lets you specify the left coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
11	.**VectorBBoxRight** (optional) Lets you specify the right coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
12	.**RotationAngle** (optional) Lets you specify the amount in which the tile is rotated in millionths of degrees.
13	.**SkewAngle** (optional) Lets you specify the amount in which the tile is skewed in millionths of degrees.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyFullColorFill "C:\MONTEMP.BMP", 500000, 500000, 100, 100, 0, 100, 0, 0
```

The above example applies a full color fill to a rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

CorelScript.ApplyTextureFill

Function **ApplyTextureFill**(ByVal **TextureLibrary** AS String, ByVal **TextureName** AS String, ByVal **TextureStyle** AS String, ByVal **TextureWidth** AS Long, ByVal **TextureHeight** AS Long, ByVal **TextureOffestX** AS Long, ByVal **TextureOffsetY** AS Long, ByVal **RowOffset** AS Boolean, ByVal **RowColumnOffset** AS Long, ByVal **ScaleWithObject** AS Boolean, ByVal **RotationAngle** AS Long, ByVal **SkewAngle** AS Long) AS Long

0 CorelScript

Description

This command lets you apply one of the texture fills included in CorelDRAW.

Parameter	Description
.**TextureLibrary**	Lets you specify the name of the Texture Library.
0	.**TextureName** Lets you specify the name of the texture.
1	.**TextureStyle** Lets you specify the name of the style. If you set .TextureLibrary to "Samples 5", the style name must be preceded by "CDR5:". For example, "CDR5:Blue Valley".
2	.**TextureWidth** If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
3	.**TextureHeight** If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
4	.**TextureOffsetX** X offset from center of object in the same units used by width & height.
5	.**TextureOffsetY** Y offset from center of object in the same units used by widht & height.
6	.**RowOffset** Set to TRUE enables the row offset, Set to FALSE enables the column offset.
7	.**RowColumnOffset** Lets you specify the amount that the row or columns is offset as a percentage.
8	.**ScaleWithObject** Set to TRUE will transform the fill with the object.
9	.**RotationAngle** Lets you specify the amount that the texture is rotated in millionths of degrees.
10	.**SkewAngle** Lets you specify the amount that the texture is skewed in millionths of degrees.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyTextureFill "Styles", "Satellite Photography", "Satellite Photography"
```

The above example creates a rectangle, then applies the satellite photography fill to it.

{button ,AL(`CLS_CorelScript;FNC_ApplyTextureFill')} **Related Topics**

CorelScript.ApplyPostscriptFill

Function **ApplyPostscriptFill**(ByVal **PSFill** AS String, ByVal **NumParms** AS Long, ByVal **Parm1** AS Long, ByVal **Parm2** AS Long, ByVal **Parm3** AS Long, ByVal **Parm4** AS Long, ByVal **Parm5** AS Long) AS Long

0 CorelScript

Description

This command lets you apply a PostScript Fill to a selected object.

Parameter	Description
.**PSFill**	Lets you specify the name of the postscript fill. The name must be preceded by an F/. For a listing of Postscript fills available, see the PostScript Texture dialog box. If you create custom PostScript fills, their definitions are placed in the USERPROC.PS file in the Custom folder of your Corel folder. The PostScript fills definitions supplied with DRAW are also in this file.
0	.**NumParms** Lets you specify the number of parameters used for the selected PostScript Fill, an integer value between 1 and 5, inclusive. Refer to the PostScript Texture dialog box to determine the number of parameters for a full fill. Set to 2 for spot fills.
1	.**Parm1** Lets you specify the first parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If you're using a spot fill, set Parm1 to a value between -1 and 1.
2	.**Parm2** Lets you specify the second parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set Parm2 to a value between -1 and 1.
3	.**Parm3** Lets you specify the third parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.
4	.**Parm4** Lets you specify the fourth parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.
5	.**Parm5** Lets you specify the fifth parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.

Note

- If you create custom PostScript fills, their definitions are placed in the USERPROC.PS file in the Custom folder of your Corel folder. The PostScript fills definitions supplied with CorelDRAW are also in this file.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyPostScriptFill "F/StoneWall", 4, 15, 100,0, 5, 0
```

The above example applies the StoneWall PostScript fill to the selected rectangle.

{button ,AL(`CLS_CorelScript;FNC_ApplyPostscriptFill')} **Related Topics**

CorelScript.ApplyOutline

Function **ApplyOutline**(ByVal **Width** AS Long, ByVal **Type** AS Long, ByVal **EndCaps** AS Long, ByVal **JoinType** AS Long, ByVal **Aspect** AS Long, ByVal **Angle** AS Long, ByVal **DotDash** AS Long, ByVal **RightArrow** AS Long, ByVal **LeftArrow** AS Long, ByVal **BehindFill** AS Boolean, ByVal **OutlineType** AS Long, ByVal **Preset** AS Long, ByVal **ScalePen** AS Boolean) AS Long

0 CorelScript

Description

This command lets you apply an Outline to the selected object.

Parameter	Description
.**Width**	Lets you specify the width of the outline to apply, in tenths of a micron.
0 .**Type**	Lets you specify the outline type: 0 = None 1 = Solid 2 = Dot - Dash
1 .**EndCaps**	Lets you specify the end caps to be applied to the outline: 0 = Butt 1 = Round 2 = Square
2 .**JoinType**	Lets you specify the outline join types: 0 = Miter 1 = Round 2 = Bevel
3 .**Aspect**	Lets you specify the stretch field which adjusts the width of the nib. Valid values range from 1 to 100 percent.
4 .**Angle**	Lets you specify the angle of the nib's edge, in tenths of degrees.
5 .**DotDash**	Lets you specify the type of dot/dash line. Dot/dash line types are listed in the Style drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list▸the first listed type is identified as 0, the second listed type is identified as 1, and so on.
6 .**RightArrow**	Lets you specify the style of right-arrow. Right-arrow types are listed in the right arrow drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list (from left-to-right)▸the first listed type is identified as 0, the second listed type is identified as 1, and so on.
7 .**LeftArrow**	Lets you specify the style of left-arrow. Left-arrow types are listed in the left arrow drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list (from left-to-right)▸the first listed type is identified as 0, the second listed type is identified as 1, and so on.
8 .**BehindFill**	Set to TRUE (-1) to position the outline behind the fill. Set to FALSE (0) to position the outline in front of the fill.
9 .**OutlineType**	Lets you specify the type of preset outline. 10 0 = Pen 1 = Outline 2 = PenOutline
11 .**Preset**	Lets you specify the tint of the outline. Values range from 1 (0%) to 11 (100%). A value of 0 has no effect on the outline. This parameter is only used if the selected outline type supports preset tints.
12 .**ScalePen**	Set to TRUE (-1) to scale the outline when the object is scaled.

Note

- You can use the ANGLECONVERT function to specify angle measurements
- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyOutline command.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyOutline 50000, 2, 0, 1, 50, 250, 2, 0, 0, 0
```

The above example applies a dashed outline 50000 microns wide, with round corners to the rectangle.

{button ,AL(`CLS_CorelScript;FNC_ApplyOutline')} **Related Topics**

CorelScript.GetOutline

Function **GetOutline**(ByRef **Width** AS Long, ByRef **Type** AS Long, ByRef **EndCaps** AS Long, ByRef **JoinType** AS Long, ByRef **Aspect** AS Long, ByRef **Angle** AS Long, ByRef **DotDash** AS Long, ByRef **RightArrow** AS Long, ByRef **LeftArrow** AS Long, ByRef **BehindFill** AS Boolean, ByRef **ScalePen** AS Boolean) AS Long

0 CorelScript

Description

This function returns the outline attributes of the selected object. If more than one object is selected, the function returns the outline attributes of the last selected object.

Parameter	Description
.**Width**	Returns the width of the outline, in tenths of a micron.
0	.**Type** Returns the outline type: 0 = None 1 = Solid 2 = Dot - Dash
1	.**EndCaps** Returns the End Caps applied to the outline: 0 = Butt 1 = Round 2 = Square
2	.**JoinType** Returns the outline join types: 0 = Miter 1 = Round 2 = Bevel
3	.**Aspect** Returns the stretch field which adjusts the width of the nib.
4	.**Angle** Returns the angle of the nib's edge, in tenths of degrees.
5	.**DotDash** Returns the type of dot/dash line. Refer to the Outline Pen dialog box for more details.
6	.**RightArrow** Returns the style of right-arrow. Refer to the Outline Pen dialog box for more details.
7	.**LeftArrow** Returns the style of left-arrow. Refer to the Outline Pen dialog box for more details.
8	.**BehindFill** Returns the position of the outline fill. TRUE (-1) = Outline behind fill FALSE (0) = Outline in front of fill
9	.**ScalePen** Returns the the scale pen setting. TRUE (-1) = Outline is scaled when object is scaled FALSE (0) = Outline is not scaled when object is scaled

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.GetOutline Width&, outlineType&, EndCaps&, JoinType&, Aspect&, Angle&, DotDash&, RArrow&,
LArrow&, BehindFill&
```

The above example returns the outline attributes of the selected object.

{button ,AL(`CLS_CorelScript;FNC_GetOutline')} **Related Topics**

CorelScript.GetOutlineColor

Function **GetOutlineColor**(ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef **V2** AS Long, ByRef **V3** AS Long, ByRef **V4** AS Long, ByRef **V5** AS Long, ByRef **V6** AS Long, ByRef **Density** AS Long) AS Long

0 CorelScript

Description

This function returns the Outline Color attributes of a selected object. If more than one object is selected, the function returns the Outline Color attributes of the last selected object.

Parameter	Description
.**ColorModel**	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome 0 To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 1 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
2	.**Color1** Returns the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
3	.**Color2** Returns the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges.
4	.**Color3** Returns the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges.
5	.**Color4** Returns the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges.
6	.**Color5** Lets you specify the fifth color component for .ColorModel.
7	.**Color6** Lets you specify the sixth color component for .ColorModel.
8	.**Density** Lets you specify the color density from 0-100.

Example

```
.GetOutlineColor Model&, C1&, C2&, C3&, C4&, C5&, C6&, C7&
MESSAGE Model&
```

The above example determines the outline color attributes of the selected object and displays a number corresponding to the color model in a message box.

{button ,AL(`CLS_CorelScript;FNC_GetOutlineColor')} **Related Topics**

CorelScript.SetCharacterAttributes

Function **SetCharacterAttributes**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **FontName** AS String, ByVal **FontStyle** AS Long, ByVal **PointSize** AS Long, ByVal **Underline** AS Long, ByVal **Overline** AS Long, ByVal **StrikeOut** AS Long, ByVal **Placement** AS Long, ByVal **CharacterSpacing** AS Long, ByVal **WordSpacing** AS Long, ByVal **LineSpacing** AS Long, ByVal **Alignment** AS Long) AS Long

0 CorelScript

Description

This command sets the text character attributes.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**FontName** Lets you specify the font name.
2	.**FontStyle** Lets you specify the style of the selected font. 7 = Normal 8 = Normal/Italic 13 = Bold 14 = Bold/Italic
3	.**PointSize** Lets you specify the size of the selected font in tenths of a point.
4	.**Underline** Lets you specify the type of underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
5	.**Overline** Lets you specify the type of overline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
6	.**StrikeOut** Lets you specify the type of strikeout. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
.**Placement**	Lets you specify the placement of the font. 0 = Normal 1 = Superscript 2 = Subscript
.**CharacterSpacing**	Lets you specify the character spacing in tenths of a percent.
0	.**WordSpacing** Lets you specify the word spacing in tenths of a percent.
1	.**LineSpacing** Lets you specify the line spacing in tenths of a percent.
2	.**Alignment** Lets you specify the alignment. 0 = None 1 = Left 2 = Center 3 = Right 4 = Full justify 5 = Force justify

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetCharacterAttributes command.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
.SelectObjectsInRect 250000, -300000, -250000, 1100000, 0
```

```
.SetCharacterAttributes 0, 4, "Arial", 13, 900, 0, 0, 0, 0, 0, 0, 1
```

The above example creates the text "COREL", then sets the font to Arial, the font type to Bold, and the point size to 90.

{button ,AL(`CLS_CorelScript;FNC_SetCharacterAttributes')} **Related Topics**

CorelScript.SetParagraphSpacing

Function **SetParagraphSpacing**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **CharacterSpacing** AS Long, ByVal **WordSpacing** AS Long, ByVal **LineSpacing** AS Long, ByVal **BeforeParagraph** AS Long, ByVal **AfterParagraph** AS Long, ByVal **Alignment** AS Long, ByVal **AutoHyphenation** AS Boolean, ByVal **HyphenHotZone** AS Long) AS Long

0 CorelScript

Description

This command sets paragraph spacing.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**CharacterSpacing** Lets you specify the character spacing in tenths of a percent.
2	.**WordSpacing** Lets you specify the word spacing in tenths of a percent.
3	.**LineSpacing** Lets you specify the line spacing in tenths of a percent.
4	.**BeforeParagraph** Lets you specify the spacing before paragraphs in tenths of a percent.
5	.**AfterParagraph** Lets you specify the spacing after paragraphs in tenths of a percent.
6	.**Alignment** Lets you specify the alignment. 0 = None 1 = Left 2 = Center 3 = Right 4 = Full justify 5 = Force justify
7	.**AutoHyphenation** Set to TRUE (-1) to enable automatic hyphenation. Set to FALSE (0) to disable this option.
8	.**HyphenHotZone** Lets you specify the size of the hyphen hot zone in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetParagraphSpacing 0, 0, 900, 900, 900, 200, 200, 1, 0, 0
```

The above example creates a text string, selects the entire text and applies paragraph spacing to it.

{button ,AL(`CLS_CorelScript;FNC_SetParagraphSpacing')} **Related Topics**

CorelScript.AddTabStop

Function **AddTabStop**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **TabStop** AS Long) AS Long

0 CorelScript

Description

This command adds tab stops to text.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**TabStop** Lets you specify the distance at which to apply tabs, in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.AddTabStop 0, 0, 1270000
```

The above example adds a tab stop every 0.5 inch.

{button ,AL(`CLS_CorelScript;FNC_AddTabStop')} **Related Topics**

CorelScript.SetIndents

Function **SetIndents**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **FirstLine** AS Long, ByVal **RestOfLines** AS Long, ByVal **RightMargin** AS Long) AS Long

0 CorelScript

Description

This command sets indents for text.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**FirstLine** Lets you specify the size of the first line indentation, in tenths of a micron.
2	.**RestOfLines** Lets you specify the size of the remaining line indentation, in tenths of a micron.
3	.**RightMargin** Lets you specify the size of the right margin indentation, in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetIndents 0, 0, 0, 400000, 0
```

The above example indents all lines except the first by 1.57 inches.

{button ,AL(`CLS_CorelScript;FNC_SetIndents')} **Related Topics**

CorelScript.SetBullet

Function **SetBullet**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **SymbolLibrary** AS String, ByVal **SymbolNumber** AS Long, ByVal **PointSize** AS Long, ByVal **BulletIndent** AS Long, ByVal **VerticalShift** AS Long) AS Long

0 CorelScript

Description

This command sets bullets for text.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**SymbolLibrary** Lets you specify the name of the symbol library. Refer to the Effects tab of the Paragraph dialog box for more details.
2	.**SymbolNumber** Lets you specify the selected symbol number. Refer to the Effects tab of the Paragraph dialog box for more details.
3	.**PointSize** Lets you specify the point size in tenths of a point.
4	.**BulletIndent** Lets you specify the size of the bullet indentation in tenths of a micron.
5	.**VerticalShift** Lets you specify the amount of baseline shift in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetBullet 32, 123, "Animals 1", 55, 480, 400000, 0
```

The above inserts a camel bullet, indented 1.57 inches.

{button ,AL(`CLS_CorelScript;FNC_SetBullet')} **Related Topics**

CorelScript.SetFrameColumn

Function **SetFrameColumn**(ByVal **ColumnNumber** AS Long, ByVal **Width** AS Long, ByVal **GutterWidth** AS Long) AS Long

0 CorelScript

Description

This command formats columns for text.

Parameter	Description
.**ColumnNumber**	Lets you specify the column number.
0	.**Width** Lets you specify the width of the column in tenths of a micron.
1	.**GutterWidth** Lets you specify the width of the gutter in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command. This command must be called twice.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetFrameColumn 0, 500000, 50000
.SetFrameColumn 1, 500000, 50000
```

The above example formats the text into two columns, each 2 inches wide.

{button ,AL(`CLS_CorelScript;FNC_SetFrameColumn')} **Related Topics**

CorelScript.CreateTextString

Function **CreateTextString**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **Text** AS String) AS Long

0 CorelScript

Description

This command creates the text.

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the text's bounding box in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the text's bounding box in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the text's bounding box in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the X-coordinate of the lower-right corner of the text's bounding box in tenths of a micron, relative to the center of the page.
3	.**Text** Lets you specify the text. Maximum string length is 255 characters.

Note

- This function must be called first to create the text before any of the functions which manipulate the text.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
```

The above example creates the text "COREL".

{button ,AL(`CLS_CorelScript;FNC_CreateTextString')} **Related Topics**

CorelScript.SetTextString

Function **SetTextString**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **Text** AS String) AS Long

0 CorelScript

Description

This command changes the text in a selected text object.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0	.**LastSelectedChar** Lets you specify the ending character of the selected text.
1	.**Text** Lets you specify the text. Maximum string length is 255 characters.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
.SelectObjectsInRect 250000, -300000, -250000, 1100000, 0
.SetCharacterAttributes 0, 4, "Arial", 13, 900, 0, 0, 0, 0, 0, 0, 0, 1
.SetTextString -1, -1, "RT"
.SetCharacterAttributes 5, 6, "Arial", 8, 900, 0, 0, 0, 1, 0, 0, 0, 0
```

The above example creates the text string "COREL", then appends a second text string "RT" to it. The appended string is italic and superscript.

{button ,AL(`CLS_CorelScript;FNC_SetTextString')} **Related Topics**

CorelScript.AlignTextToBaseline

Function **AlignTextToBaseline**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long) AS Long

0 CorelScript

Description

This command aligns text to the baseline.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0 .**LastSelectedChar**	Lets you specify the ending character of the selected text.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of underline.
0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double
thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.AlignTextToBaseline 0, 0
```

The above example aligns the text to the baseline.

{button ,AL(`CLS_CorelScript;FNC_AlignTextToBaseline')} **Related Topics**

CorelScript.StraightenText

Function **StraightenText**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long) AS Long

0 CorelScript

Description

This command resets the kerning angle to 0 for selected text.

Parameter	Description
.**FirstSelectedChar**	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
0 .**LastSelectedChar**	Lets you specify the ending character of the selected text.

Example

```
.StraightenText 0, 0
```

The above example straightens the first character of selected paragraph text.

{button ,AL(`CLS_CorelScript;FNC_StraightenText')} **Related Topics**

CorelScript.InsertOLEObject

Function **InsertOLEObject**(ByVal **ProgID** AS String) AS Long

0 CorelScript

Description

This command inserts an OLE object in a CorelDRAW document.

Parameter	Description
.**ProgID**	Lets you specify the OLE object's Windows registry name.

Example

```
.InsertOLEObject "CorelPhotoPaint.Image.6"
```

The above example inserts a Corel PHOTO-PAINT image into a CorelDRAW document.

{button ,AL(`CLS_CorelScript;FNC_InsertOLEObject')} **Related Topics**

CorelScript.InsertOLEObjectFromFile

Function **InsertOLEObjectFromFile**(ByVal **FileName** AS String, ByVal **CreateLink** AS Boolean) AS Long

0 CorelScript

Description

This command inserts an OLE object from a file into a CorelDRAW document.

Parameter	Description
.**FileName**	The filename.
0 .**CreateLink**	Set to TRUE (-1) to create a link. Set to FALSE (0) to disable this option.

Example

```
.InsertOLEObjectFromFile "C:\WINWORD\WORDFILE.DOC", -1
```

The above example inserts a Microsoft Word file in a CorelDRAW document.

{button ,AL(`CLS_CorelScript;FNC_InsertOLEObjectFromFile')} **Related Topics**

CorelScript.OLEObjectDoVerb

Function **OLEObjectDoVerb**(ByVal **Verb** AS Long) AS Long

0 CorelScript

Description

This command performs the specified action on an OLE object.

Parameter	Description
.**Verb**	Lets you specify the OLE object action to perform. 0 = Primary 1 = Secondary 2 = Tertiary etc.

Note

- Primary and secondary verbs depend on the object type.

Example

```
.InsertOLEObject "CorelPhotoPaint.Image.7"
.OLEObjectDoVerb 0
```

The above example inserts a Corel PHOTO-PAINT OLE object into a DRAW document and invokes in-place editing.

{button ,AL(`CLS_CorelScript;FNC_OLEObjectDoVerb')} **Related Topics**

CorelScript.PasteSystemClipboardFormat

Function **PasteSystemClipboardFormat**(ByVal **Format** AS Long) AS Long

0 CorelScript

Description

This command specifies the system format for pasting from the Clipboard.

Parameter	Description
.**Format**	Lets you specify the type of format. 1 = CF Text 2 = Bitmap 3 = Metafile Pict 8 = DIB 14 = Enhanced Metafile

Example

```
.PasteSystemClipboardFormat 2
```

The above example pastes a bitmap from the Clipboard into the active document.

{button ,AL(`CLS_CorelScript;FNC_PasteSystemClipboardFormat')} **Related Topics**

CorelScript.PasteCustomClipboardFormat

Function **PasteCustomClipboardFormat**(ByVal **Format** AS String) AS Long

0 CorelScript

Description

This command specifies the custom format for pasting from the Clipboard.

Parameter	Description
.**Format**	Lets you specify the type of format. Options include: "Corel 32-bit Presentation Exchange Data" "Corel Presentation Exchange Data" "Corel Metafile" "Rich Text Format"

Example

```
.PasteCustomClipboardFormat "Rich Text Format"
```

The above example inserts the contents of the Clipboard into a CorelDRAW document as Rich Text.

{button ,AL(`CLS_CorelScript;FNC_PasteCustomClipboardFormat')} **Related Topics**

CorelScript.FindObjectOfStyle

Function **FindObjectOfStyle**(ByVal **StyleName** AS String) AS Boolean

0 CorelScript

Description

This function finds the next object with the specified style.

Return Value

Returns one of the following values:

- TRUE (-1) an object is found
- FALSE (0) no object is found.

Parameter	Description
.**StyleName**	Lets you specify the name of the style.

{button ,AL(`CLS_CorelScript;FNC_FindObjectOfStyle')} **Related Topics**

CorelScript.FindNextObjectOfStyle

Function **FindNextObjectOfStyle**() AS Boolean

0 CorelScript

Description

This function finds the next object with the current style.

Return Value

Returns one of the following values

- TRUE (-1) an object is found
- FALSE (0) no object is found.

{button ,AL(`CLS_CorelScript;FNC_FindNextObjectOfStyle')} **Related Topics**

CorelScript.SaveStyleAs

Function **SaveStyleAs**(ByVal **Style** AS String, ByVal **Fill** AS Boolean, ByVal **Outline** AS Boolean, ByVal **TypeFace** AS Boolean, ByVal **TypeStyle** AS Boolean, ByVal **Size** AS Boolean, ByVal **Justify** AS Boolean, ByVal **Tabs** AS Boolean, ByVal **Hyphenation** AS Boolean, ByVal **SpaceChar** AS Boolean, ByVal **SpaceWord** AS Boolean, ByVal **SpaceLine** AS Boolean, ByVal **BeforePara** AS Boolean, ByVal **AfterPara** AS Boolean, ByVal **Underline** AS Boolean, ByVal **Overline** AS Boolean, ByVal **StrikeOut** AS Boolean, ByVal **BulletIndent** AS Boolean, ByVal **FirstLineIndent** AS Boolean, ByVal **RestOfLinesIndent** AS Boolean, ByVal **RightMargin** AS Boolean, ByVal **SuperOrSubScript** AS Boolean, ByVal **Capitalize** AS Boolean, ByVal **Bullet** AS Boolean) AS Long

0 CorelScript

Description

This command saves the style of the current object as a new style.

Parameter	Description
.**Style**	Lets you specify the name of the new style.
0	.**Fill** Set to TRUE (-1) to include fill properties. Set to FALSE (0) to exclude these properties.
1	.**Outline** Set to TRUE (-1) to include outline properties. Set to FALSE (0) to exclude these properties.
2	.**Typeface** Set to TRUE (-1) to include typeface properties. Set to FALSE (0) to exclude these properties.
3	.**TypeStyle** Set to TRUE (-1) to include type style properties. Set to FALSE (0) to exclude these properties.
4	.**Size** Set to TRUE (-1) to include size properties. Set to FALSE (0) to exclude these properties.
5	.**Justification** Set to TRUE (-1) to include text justification properties. Set to FALSE (0) to exclude these properties.
6	.**Tabs** Set to TRUE (-1) to include tab stop properties. Set to FALSE (0) to exclude these properties.
7	.**Hyphenation** Set to TRUE (-1) to include hyphenation properties. Set to FALSE (0) to exclude these properties.
8	.**SpaceChar** Set to TRUE (-1) to include character spacing properties. Set to FALSE (0) to exclude these properties.
9	.**SpaceWord** Set to TRUE (-1) to include word spacing properties. Set to FALSE (0) to exclude these properties.
10	.**SpaceLine** Set to TRUE (-1) to include line spacing properties. Set to FALSE (0) to exclude these properties.
11	.**BeforePara** Set to TRUE (-1) to include paragraph spacing properties (before the paragraph). Set to FALSE (0) to exclude these properties.
12	.**AfterPara** Set to TRUE (-1) to include paragraph spacing properties (after the paragraph). Set to FALSE (0) to exclude these properties.
13	.**Underline** Set to TRUE (-1) to include text underline properties. Set to FALSE (0) to exclude these properties.
14	.**Overline** Set to TRUE (-1) to include text overline properties. Set to FALSE (0) to exclude these properties.
15	.**Strikeout** Set to TRUE (-1) to include text strikeout properties. Set to FALSE (0) to exclude these properties.
16	.**BulletIndent** Set to TRUE (-1) to include bullet indentation properties. Set to FALSE (0) to exclude these properties.
17	.**FirstLineIndent** Set to TRUE (-1) to include indentation properties for the first line. Set to FALSE (0) to exclude these properties.
18	.**RestOfLinesIndent** Set to TRUE (-1) to include indentation properties for remaining lines (hanging indent). Set to FALSE (0) to exclude these properties.
19	.**RightMargin** Set to TRUE (-1) to include right margin properties. Set to FALSE (0) to exclude these properties.
20	.**SuperOrSubScript** Set to TRUE (-1) to include superscript or subscript properties. Set to FALSE (0) to exclude these properties.
21	.**Capitalize** Set to TRUE (-1) to include capitalization properties. Set to FALSE (0) to exclude these properties.
22	.**Bullet** Set to TRUE (-1) to include bullet properties. Set to FALSE (0) to exclude these properties.

{button ,AL(`CLS_CorelScript;FNC_SaveStyleAs')} **Related Topics**

CorelScript.RevertToStyle

Function **RevertToStyle**() AS Long

0 CorelScript

Description

This command converts an object to its original style.

{button ,AL(`CLS_CorelScript;FNC_RevertToStyle')} **Related Topics**

CorelScript.RedrawScreen

Function **RedrawScreen**() AS Long

0 CorelScript

Description

This command forces CorelDRAW to redraw the windows of the active document.

{button ,AL(`CLS_CorelScript;FNC_RedrawScreen')} **Related Topics**

CorelScript.RedrawAllScreens

Function **RedrawAllScreens**() AS Long

0 CorelScript

Description

This command forces CorelDRAW to redraw all open document windows.

{button ,AL(`CLS_CorelScript;FNC_RedrawAllScreens')} **Related Topics**

CorelScript.StretchObject

Function **StretchObject**(ByVal **XScaleNumerator** AS Double, ByVal **XScaleDenominator** AS Double, ByVal **YScaleNumerator** AS Double, ByVal **YScaleDenominator** AS Double, ByVal **HMirror** AS Boolean, ByVal **VMirror** AS Boolean, ByVal **ReferenceNum** AS Long) AS Long

0 CorelScript

Description

This command stretches or mirrors the selected object.

Parameter	Description
.**XScaleNumerator**	Lets you specify the amount that the selected object is stretched along the X-axis. The final stretch value is determined by dividing this number by .XScaleDenominator. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
0	.**XScaleDenominator** Lets you specify the amount that the selected object is stretched along the X-axis. The final stretch value is determined by dividing .XScaleNumerator by this number. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
1	.**YScaleNumerator** Lets you specify the amount that the selected object is stretched along the Y-axis. The final stretch value is determined by dividing this number by .YScaleDenominator. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
2	.**YScaleDenominator** Lets you specify the amount that the selected object is stretched along the Y-axis. The final stretch value is determined by dividing .YScaleNumerator by this number. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
3	.**HMirror** Set to TRUE (-1) to horizontally mirror the selected object.
4	.**VMirror** Set to TRUE (-1) to vertically mirror the selected object.
5	.**ReferenceNum** Lets you specify the reference point of the object to be skewed. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

{button ,AL(`CLS_CorelScript;FNC_StretchObject')} **Related Topics**

CorelScript.OverprintOutline

Function **OverprintOutline**() AS Long

0 CorelScript

Description

This command overprints the outline of the selected object.

{button ,AL(`CLS_CorelScript;FNC_OverprintOutline')} **Related Topics**

CorelScript.OverprintFill

Function **OverprintFill**() AS Long

0 CorelScript

Description

This command overprints the fill of the selected object.

{button ,AL(`CLS_CorelScript;FNC_OverprintFill')} **Related Topics**

CorelScript.CopyPropertiesFrom

Function **CopyPropertiesFrom**(ByVal **FromObjectID** AS Long, ByVal **OutlinePen** AS Boolean, ByVal **OutlineColor** AS Boolean, ByVal **Fill** AS Boolean, ByVal **TextAttributes** AS Boolean) AS Long

0 CorelScript

Description

This command copies the properties from the object with the specified object ID to the selected object.

Parameter	Description
.**FromObjectID**	Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.
0	.**OutlinePen** Set to TRUE (-1) to copy outline pen properties. Set to FALSE (0) to exclude outline pen properties.
1	.**OutlineColor** Set to TRUE (-1) to copy outline color properties. Set to FALSE (0) to exclude outline color properties.
2	.**Fill** Set to TRUE (-1) to copy fill properties. Set to FALSE (0) to exclude fill properties.
3	.**TextAttibutes** Set to TRUE (-1) to copy text properties. Set to FALSE (0) to exclude text properties.

{button ,AL(`CLS_CorelScript;FNC_CopyPropertiesFrom')} **Related Topics**

CorelScript.FitTextToPath

Function **FitTextToPath**(ByVal **TextOrientation** AS Long, ByVal **VertAlign** AS Long, ByVal **HorizAlign** AS Long, ByVal **CurveSideToFit** AS Long, ByVal **FitOtherSide** AS Boolean, ByVal **HorizOffset** AS Long, ByVal **DistFromPath** AS Long) AS Long

0 CorelScript

Description

This command fits selected artistic text to the selected path.

Parameter	Description
.**TextOrientation**	0 = Rotates individual characters to follow the contours of the path. 1 = The characters are not changed. 2 = Vertically skews each character, creating the impression that the text is standing upright on the path. 3 = Horizontally skews each character, creating the impression that the text is turning in toward the screen.
0	.**VertAlign** If you specify a distamce from the path, then this parameter has no effect.
1	0 = Variable. Allows you to move the text off the path by dragging with the mouse. 1 = Bottom. Aligns the descender line of the text with the path. 2 = Top. Aligns the ascender line of the text with the path. 3 = Center. Centers the text vertically on the path. 4 = Baseline. Aligns the baseline of the text with the path.
2	.**HorizAlign** 1 = Aligns the text with the start node of the line or curve. 2 = Aligns the text with the end point of the line or curve. 3 = Centers the text on the path.
3	.**CurveSideToFit** 1 = Aligns the text to the top of a closed object. 2 = Aligns the text to the left of a closed object. 3 = Aligns the text to the bottom of a closed object. 4 = Aligns the text to the right of a closed object.
4	.**FitOtherSide** Set to TRUE (-1) to place the text on the other side of the path.
5	.**HorizOffset** Lets you specify the distance the text is offset from the start node.
6	.**DistFromPath** Lets you specify the distance the text is from path.

{button ,AL(`CLS_CorelScript;FNC_FitTextToPath')} **Related Topics**

CorelScript.ApplyContour

Function **ApplyContour**(ByVal **ContourType** AS Long, ByVal **Offset** AS Long, ByVal **Steps** AS Long, ByVal **ColorWheelDirection** AS Long) AS Long

0 CorelScript

Description

This command applies a contour to the selected object.

Parameter	Description
.**ContourType**	0 = To Center 1 = Inside 2 = Outside
0	.**Offset** Lets you specify the distance between contours in tenths of a micron.
1	.**Steps** Lets you specify the number of steps.
2	.**ColorWheelDirection** 0 = straight 1 = clockwise 2 = counter-clockwise

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyContour command.

{button ,AL(`CLS_CorelScript;FNC_ApplyContour')} **Related Topics**

CorelScript.ApplyEnvelopeFrom

Function **ApplyEnvelopeFrom**(ByVal **ObjectID** AS Long, ByVal **MappingMode** AS Long, ByVal **KeepLines** AS Boolean) AS Long

0 CorelScript

Description

This command applies an envelope to the selected object from the shape of another object.

Parameter	Description
.**ObjectID**	Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.
0	.**Mappingmode** 0 = Horizontal 1 = Original 2 = Putty 3 = Vertical
1	.**KeepLines** Set to TRUE (-1) to keep the lines of the source object. Set to FALSE (0) to exclude these lines.

{button ,AL(`CLS_CorelScript;FNC_ApplyEnvelopeFrom')} **Related Topics**

CorelScript.ApplyPresetEnvelope

Function **ApplyPresetEnvelope**(ByVal **PresetNumber** AS Long, ByVal **MappingMode** AS Long, ByVal **KeepLines** AS Boolean) AS Long

0 CorelScript

Description

This command applies a preset envelope to the selected object.

Parameter	Description
.**PresetNumber**	Lets you specify the preset envelope to use. Refer to the Preset Roll-Up to see which presets are available. Valid values range from 1 to 39.
0	.**Mappingmode** 0 = Horizontal 1 = Original 2 = Putty 3 = Vertical
1	.**KeepLines** Set to TRUE (-1) to keep the lines of the envelope. Set to FALSE (0) to exclude these lines.

{button ,AL(`CLS_CorelScript;FNC_ApplyPresetEnvelope')} **Related Topics**

CorelScript.ApplyBlend

Function **ApplyBlend**(ByVal **Steps** AS Boolean, ByVal **NoOfStepsSpaces** AS Long, ByVal **AngleOfRotation** AS Long, ByVal **Loop** AS Boolean, ByVal **PathObjectID** AS Long, ByVal **FullPath** AS Boolean, ByVal **RotateAll** AS Boolean, ByVal **ColorWheelMode** AS Long, ByVal **MapNodeStartObject** AS Long, ByVal **MapNodeEndObject** AS Long, ByVal **LinearBlend** AS Boolean, ByVal **LinearSpacing** AS Boolean, ByVal **LinkAccelerations** AS Boolean, ByVal **AccelShapes** AS Boolean, ByVal **BlendLogBase** AS Long, ByVal **SpacingLogBase** AS Long, ByVal **BlendID** AS Long, ByVal **BlendType** AS Long, ByVal **MapNodes** AS Boolean) AS Long

0 CorelScript

Description

This command applies a blend to two selected objects. The parameters below correspond to the controls in the Blend Roll-Up.

Parameter	Description
.**Steps**	Set to TRUE (-1) to set the number of steps. Set to FALSE (0) if the blend is on a path and you want to use fixed spacing along that path.
0	.**NoOfSteps** Lets you specify the number of intermediate steps.
1	.**AngleOfRotation** Lets you specify the rotation of the intermediate steps in millionths of a degree (e.g., 5000000 = 5 degrees).
2	.**Loop** Set to TRUE (-1) to enable the loop option. Set to FALSE (0) to disable the loop option.
3	.**PathObjectID** Lets you specify the object ID of the path object. Use .GetObjectsCDRStaticID to get an object's ID.
4	.**FullPath** Set to TRUE (-1) to enable the blend along full path option. Set to FALSE (0) to disable the blend along full path option.
5	.**RotateAll** Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
6	.**ColorWheelMode** 0 = straight 1 = clockwise 2 = counter-clockwise
7	.**MapNodeStartObject** Lets you specify a node on the start object to map to a specific node on the end object. The value can range from 0 (the first node) to the number of nodes minus 1.
8	.**MapNodeEndObject** Lets you specify a node on the end object to map to a specific node on the start object. The value can range from 0 (the first node) to the number of nodes minus 1.
9	.**LinearBlend** Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
10	.**LinearSpacing** Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
11	.**LinkAcceleration** Set to TRUE (-1) to link the blend acceleration options.
12	.**AccelShapes** Set to TRUE (-1) to accelerate the change in size between the start and end objects.
13	.**BendLogBase** Lets you specify the rate of color acceleration.
14	.**SpacingLogBase** Lets you specify the rate of spacing acceleration.
15	.**BlendID** Reserved for future use.
16	.**BlendType** Reserved for future use.

{button ,AL(`CLS_CorelScript;FNC_ApplyBlend')} **Related Topics**

CorelScript.SplitBlend

Function **SplitBlend**(ByVal **PositionX** AS Long, ByVal **PositionY** AS Long) AS Long

0 CorelScript

Description

This command splits the selected blend group.

Parameter	Description
.**PositionX**	Lets you specify the X-coordinate of the point where you want the blend to be split, in tenths of a micron, relative to the center of the page.
0	.**PositionY** Lets you specify the Y-coordinate of the point where you want the blend to be split, in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_SplitBlend')} **Related Topics**

CorelScript.FuseBlend

Function **FuseBlend**(ByVal **End** AS Boolean, ByVal **PositionX** AS Long, ByVal **PositionY** AS Long) AS Long

0 CorelScript

Description

This command fuses a split blend group.

Parameter	Description
.**End**	Set to TRUE (-1) to fuse the top of the blend. Set to FALSE (0) to fuse the bottom of the blend.
0	.**PositionX** Lets you specify the X-coordinate of the point where you want the blend to be fused, in tenths of a micron, relative to the center of the page.
1	.**PositionY** Lets you specify the Y-coordinate of the point where you want the blend to be fused, in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_FuseBlend')} **Related Topics**

CorelScript.DetachBlendPath

Function **DetachBlendPath**() AS Long

0 CorelScript

Description

This command detaches a path from a blend group.

{button ,AL(`CLS_CorelScript;FNC_DetachBlendPath')} **Related Topics**

CorelScript.AppendObjectToSelection

Function **AppendObjectToSelection**(ByVal **ObjectID** AS Long) AS Boolean

0 CorelScript

Description

This command adds the object with the specified object ID to the existing selection.

Parameter	Description
.**ObjectID**	Lets you specify the object ID of the object to append. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`CLS_CorelScript;FNC_AppendObjectToSelection')} **Related Topics**

CorelScript.ClearEffect

Function **ClearEffect**(ByVal **LeaveTarget** AS Boolean) AS Long

0 CorelScript

Description

This command removes a special effect from the selected object.

Parameters	Description
LeaveTarget	Description of 'LeaveTarget' goes here

{button ,AL(`CLS_CorelScript;FNC_ClearEffect')} **Related Topics**

CorelScript.ApplyExtrude

Function **ApplyExtrude**(ByVal **ExtrudeType** AS Long, ByVal **VPProperties** AS Long, ByVal **CopyObjectID** AS Long, ByVal **Depth** AS Long, ByVal **VPHorizPos** AS Long, ByVal **VPVertPos** AS Long, ByVal **PageOrigin** AS Boolean, ByVal **Light1Pos** AS Long, ByVal **Light1Intensity** AS Long, ByVal **Light2Pos** AS Long, ByVal **Light2Intensity** AS Long, ByVal **Light3Pos** AS Long, ByVal **Light3Intensity** AS Long, ByVal **FillType** AS Long, ByVal **DrapeFill** AS Boolean) AS Long

0 CorelScript

Description

This command extrudes the selected object.

Parameter	Description
.**ExtrudeType**	0 = Small Back 1 = Small Front 2 = Big Back 3 = Big Front 4 = Back Parallel 5 = Front Parallel
0	.**VPProperties** 0 = Vanishing Point locked to object 1 = Vanishing Point locked to page 2 = Copy VP from object (specified with .lCopyObjectID) 3 = Shared VP (specified with .lCopyObjectID)
1	.**CopyObjectID** Lets you specify the object ID of the source object for shared and copied vanishing points. Use .GetObjectsCDRStaticID to get an object's ID.
2	.**Depth** Lets you specify the depth of the extrusion.
3	.**VPHorizPos** Lets you specify the X-coordinate of the vanishing point in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
4	.**VPVertPos** Lets you specify the Y-coordinate of the vanishing point in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
5	.**PageOrigin** Set to TRUE (-1) to position the vanishing point using absolute page coordinates. Set to FALSE (0) to position the vanishing point relative to the object's position.
6	.**Light1Pos** Lets you specify the position of the light source. Valid values range from 0 to 16.
7	.**Light1Intensity** Lets you specify the intensity of the light source. Valid values range from 0 to 100.
8	.**Light2Pos** Lets you specify the position of the light source. Valid values range from 0 to 16.
9	.**Light2Intensity** Lets you specify the intensity of the light source. Valid values range from 0 to 100.
10	.**Light3Pos** Lets you specify the position of the light source. Valid values range from 0 to 16.
11	.**Light3Intensity** Lets you specify the intensity of the light source. Valid values range from 0 to 100.
12	.**FillType** 0 = Object fill 1 = Solid fill 2 = Shade
13	.**DrapeFill** Set to TRUE will drape the fill over the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyExtrude')} **Related Topics**

CorelScript.ApplyRotatedExtrude

Function **ApplyRotatedExtrude**(ByVal **ExtrudeType** AS Long, ByVal **Depth** AS Long, ByVal **XRotation** AS Long, ByVal **YRotation** AS Long, ByVal **ZRotation** AS Long, ByVal **VPHorizPos** AS Long, ByVal **VPVertPos** AS Long, ByVal **PageOrigin** AS Boolean, ByVal **Light1Pos** AS Long, ByVal **Light1Intensity** AS Long, ByVal **Light2Pos** AS Long, ByVal **Light2Intensity** AS Long, ByVal **Light3Pos** AS Long, ByVal **Light3Intensity** AS Long, ByVal **FillType** AS Long, ByVal **DrapeFill** AS Boolean) AS Long

0 CorelScript

Description

This command applies a rotated extrusion to the selected object.

Parameter		Description
.**ExtrudeType**		0 = Small Back 1 = Small Front 2 = Big Back 3 = Big Front 4 = Back Parallel 5 = Front Parallel
0	.**Depth**	Lets you specify the depth of the extrusion.
1	.**XRotation**	Lets you specify the rotation value for the X-axis. Valid values range from 0 to 100.
2	.**YRotation**	Lets you specify the rotation value for the Y-axis. Valid values range from 0 to 100.
3	.**ZRotation**	Lets you specify the rotation value for the Z-axis. Valid values range from 0 to 100.
4	.**VPHorizPos**	Lets you specify the X-coordinate of the vanishing point, in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
5	.**VPVertPos**	Lets you specify the Y-coordinate of the vanishing point, in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
6	.**PageOrigin**	Set to TRUE (-1) to position the vanishing point using absolute page coordinates. Set to FALSE (0) to position the vanishing point relative to the object's position.
7	.**Light1Pos**	Lets you specify the position of the light source. Valid values range from 0 to 16.
8	.**Light1Intensity**	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
9	.**Light2Pos**	Lets you specify the position of the light source. Valid values range from 0 to 16.
10	.**Light2Intensity**	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
11	.**Light3Pos**	Lets you specify the position of the light source. Valid values range from 0 to 16.
12	.**Light3Intensity**	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
13	.**FillType**	0 = Object fill 1 = Solid fill 2 = Shade
14	.**DrapeFill**	Set to TRUE will drape the fill over the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyRotatedExtrude')} **Related Topics**

CorelScript.RecorderSelectObjectByIndex

Function **RecorderSelectObjectByIndex**(ByVal **ClearFirst** AS Boolean, ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderSelectObjectByIndex')} **Related Topics**

CorelScript.RecorderSelectObjectsByIndex

Function **RecorderSelectObjectsByIndex**(ByVal **ClearFirst** AS Boolean, ByVal **Index1** AS Long, ByVal **Index2** AS Long, ByVal **Index3** AS Long, ByVal **Index4** AS Long, ByVal **Index5** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderSelectObjectsByIndex')} **Related Topics**

CorelScript.RecorderSelectPreselectedObjects

Function **RecorderSelectPreselectedObjects**(ByVal **ClearFirst** AS Boolean) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderSelectPreselectedObjects')} **Related Topics**

CorelScript.RecorderStorePreselectedObjects

Function **RecorderStorePreselectedObjects**(ByVal **ConvertPreset** AS Boolean) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderStorePreselectedObjects')} **Related Topics**

CorelScript.StartOfRecording

Function **StartOfRecording**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_StartOfRecording')} **Related Topics**

CorelScript.EndOfRecording

Function **EndOfRecording**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_EndOfRecording')} **Related Topics**

CorelScript.SuppressPainting

Function **SuppressPainting**(ByVal **ShowDialog** AS Boolean) AS Long

0 CorelScript

Description

This command instructs CorelDRAW to suppress screen updating. To resume screen updating, use the .ResumePainting command.

Parameter	Description
.**ShowDialog**	Set to TRUE (-1) displays the script running dialog box. Set to FALSE (0) disables the script running message dialog box.

{button ,AL(`CLS_CorelScript;FNC_SuppressPainting')} **Related Topics**

CorelScript.ResumePainting

Function **ResumePainting**() AS Long

0 CorelScript

Description

This command instructs CorelDRAW to resume screen updating. To stop screen updating, use the .SupressPainting command.

{button ,AL(`CLS_CorelScript;FNC_ResumePainting')} **Related Topics**

CorelScript.CopyEffectFrom

Function **CopyEffectFrom**(ByVal **Clone** AS Boolean, ByVal **SourceObjectID** AS Long) AS Long

0 CorelScript

Description

This command copies an effect from a specific object to the selected object.

Parameter	Description
.**Clone**	Set to TRUE (-1) to clone the effect instead of copying it. This creates a link between the two objects. When the effect is changed for one object, the other object also changes.
0	.**SourceObjectID** Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`CLS_CorelScript;FNC_CopyEffectFrom')} **Related Topics**

CorelScript.BeforeObject

Function **BeforeObject**(ByVal **CorelDRAWID** AS Long) AS Long

0 CorelScript

Description

This command selects the object that is before the reference object in the object tree. Use .lObjectID to specify the reference object.

Parameter	Description
.**ObjectID**	Lets you specify the object ID of the reference object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`CLS_CorelScript;FNC_BeforeObject')} **Related Topics**

CorelScript.AfterObject

Function **AfterObject**(ByVal **CorelDRAWID** AS Long) AS Long

0 CorelScript

Description

This command selects the object that is after the reference object in the object tree. Use .lObjectID to specify the reference object.

Parameter	Description
.**ObjectID**	Lets you specify the object ID of the reference object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`CLS_CorelScript;FNC_AfterObject')} **Related Topics**

CorelScript.ApplyPerspectiveEffect

Function **ApplyPerspectiveEffect**(ByVal **Handle** AS Long, ByVal **PosX** AS Long, ByVal **PosY** AS Long) AS Long

0 CorelScript

Description

This command adds a perspective effect to the selected object.

Parameter	Description
.**Handle**	Lets you specify the handle of the object that is repositioned to create the perspective effect. Valid values range from 1 to 5.
0 .**PosX**	Lets you specify the X-coordinate of the new position of the handle, in tenths of a micron
1 .**PosY**	Lets you specify the Y-coordinate of the new position of the handle, in tenths of a micron

---

{button ,AL(`CLS_CorelScript;FNC_ApplyPerspectiveEffect')} **Related Topics**

CorelScript.ApplyLensEffect

Function **ApplyLensEffect**(ByVal **LensType** AS Long, ByVal **Frozen** AS Boolean, ByVal **RemoveFace** AS Boolean, ByVal **ViewPoint** AS Boolean, ByVal **VPX** AS Long, ByVal **VPY** AS Long, ByVal **Param1** AS Long) AS Long

0 CorelScript

Description

This command adds a lens to the selected object.

Parameter		Description
.**LensType**		0 = No Lens Effect 1 = Brighten 2 = Color Add 3 = Color Limit 4 = Custom Color Map 5 = Fish Eye 6 = Heat Map 7 = Invert 8 = Magnify 9 = Tinted Grayscale 10 = Transparancy 11 = Wireframe
0	.**Frozen**	Set to TRUE (-1) to enable the Frozen option.
1	.**RemoveFace**	Set to TRUE (-1) to enable the Remove Face option.
2	.**ViewPoint**	Set to TRUE (-1) to enable the View Point option.
3	.**VpX**	Lets you specify the X-coordinate of the view point in tenths of a micron.
4	.**VpY**	Lets you specify the Y-coordinate of the view point in tenths of a micron.
5	.**Param1**	This value will vary depending on the selected lens. Refer to the Lens Roll-Up for more information

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyLensEffect command.

{button ,AL(`CLS_CorelScript;FNC_ApplyLensEffect')} **Related Topics**

CorelScript.MoveCenter

Function **MoveCenter**(ByVal **AnchorID** AS Long, ByVal **XOffset** AS Long, ByVal **YOffset** AS Long) AS Long

0 CorelScript

Description

This command moves the selected object's center of rotation.

Parameter	Description
.**AnchorID**	Lets you specify the reference point of the object to be skewed. -1 = Use .XOffset and YOffset 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center
0	.**XOffset** If .lAnchorID = -1, then .XOffset specifies the distance, in tenths of a micron, that the center of rotation is horizontally offset from the object's center.
1	.**YOffset** If .lAnchorID = -1, then .YOffset specifies the distance, in tenths of a micron, that the center of rotation is vertically offset from the object's center.

{button ,AL(`CLS_CorelScript;FNC_MoveCenter')} **Related Topics**

CorelScript.ResetTransfo

Function **ResetTransfo**() AS Long

0 CorelScript

Description

This command clears all tranformations.

{button ,AL(`CLS_CorelScript;FNC_ResetTransfo')} **Related Topics**

CorelScript.RecorderBeginEditText

Function **RecorderBeginEditText**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderBeginEditText')} **Related Topics**

CorelScript.RecorderEditTextChangeCase

Function **RecorderEditTextChangeCase**(ByVal **CaseID** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditTextChangeCase')} **Related Topics**

CorelScript.RecorderEndEditText

Function **RecorderEndEditText**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEndEditText')} **Related Topics**

CorelScript.RecorderEditTextCharAttributes

Function **RecorderEditTextCharAttributes**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **FontName** AS String, ByVal **FontStyle** AS Long, ByVal **PointSize** AS Long, ByVal **Underline** AS Long, ByVal **Overline** AS Long, ByVal **StrikeOut** AS Long, ByVal **Placement** AS Long, ByVal **Effect** AS Long, ByVal **CharacterSpacing** AS Long, ByVal **WordSpacing** AS Long, ByVal **LineSpacing** AS Long, ByVal **Alignment** AS Long, ByVal **Direction** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditTextCharAttributes')} **Related Topics**

CorelScript.RecorderBeginEditParaText

Function **RecorderBeginEditParaText**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderBeginEditParaText')} **Related Topics**

CorelScript.RecorderEndEditParaText

Function **RecorderEndEditParaText**() AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEndEditParaText')} **Related Topics**

CorelScript.RecorderEditParaTextChangeCase

Function **RecorderEditParaTextChangeCase**(ByVal **CaseID** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditParaTextChangeCase')} **Related Topics**

CorelScript.RecorderEditParaTextSpacing

Function **RecorderEditParaTextSpacing**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **CharacterSpacing** AS Long, ByVal **WordSpacing** AS Long, ByVal **LineSpacing** AS Long, ByVal **BeforeParagraph** AS Long, ByVal **AfterParagraph** AS Long, ByVal **Alignment** AS Long, ByVal **AutoHyphenation** AS Boolean, ByVal **HyphenHotZone** AS Long, ByVal **Direction** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditParaTextSpacing')} **Related Topics**

CorelScript.RecorderEditParaTextIndents

Function **RecorderEditParaTextIndents**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **FirstLine** AS Long, ByVal **RestOfLines** AS Long, ByVal **RightMargin** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditParaTextIndents')} **Related Topics**

CorelScript.RecorderEditParaTextCharAttributes

Function **RecorderEditParaTextCharAttributes**(ByVal **FirstSelectedChar** AS Long, ByVal **LastSelectedChar** AS Long, ByVal **FontName** AS String, ByVal **FontStyle** AS Long, ByVal **PointSize** AS Long, ByVal **Underline** AS Long, ByVal **Overline** AS Long, ByVal **StrikeOut** AS Long, ByVal **Placement** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditParaTextCharAttributes')} **Related Topics**

CorelScript.RecorderEditReplaceText

Function **RecorderEditReplaceText**(ByVal **String** AS String) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditReplaceText')} **Related Topics**

CorelScript.RecorderEditParaReplaceText

Function **RecorderEditParaReplaceText**(ByVal **String** AS String) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderEditParaReplaceText')} **Related Topics**

CorelScript.MenuCommand

Function **MenuCommand**(ByVal **MenuID** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_MenuCommand')} **Related Topics**

CorelScript.ClickedDialogButton

Function **ClickedDialogButton**(ByVal **DialogID** AS Long, ByVal **ButtonID** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_ClickedDialogButton')} **Related Topics**

CorelScript.CreateSymPolygon

Function **CreateSymPolygon**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **Sides** AS Long, ByVal **Subpaths** AS Long, ByVal **Complexity** AS Long, ByVal **Star** AS Boolean, ByVal **StarComplexity** AS Long, ByVal **MaxComplexity** AS Long) AS Long

0 CorelScript

Description

This command creates a polygon or star.

Parameter	Description
.**Top**	Lets you specify the coordinate of the top of the shape in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the coordinate of the left of the shape in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the coordinate of the bottom of the shape in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the coordinate of the right of the shape in tenths of a micron, relative to the center of the page.
3	.**Sides** Lets you specify the number of sides (from 3 to 500).
4	.**Subpaths** Lets you specify the number of subpaths in a polygon. If both the number of subpaths and the complexity are set to 1 then the polygon will be a simple polygon. If either of these values are greater than 1 then the polygon becomes a star. The relationship between the number of subpaths and the complexity is represented by the Star/Polygon button and the Sharpness slider on the Polygon Property Bar. Appropriate values for each of these parameters change depending on the number of sides of the polygon.
5	.**Complexity** Lets you specify the complexity of a polygon. If both the number of subpaths and the complexity are set to 1 then the polygon will be a simple polygon. If either of these values are greater than 1 then the polygon becomes a star. The relationship between the number of subpaths and the complexity is represented by the Star/Polygon button and the Sharpness slider on the Polygon Property Bar. Appropriate values for each of these parameters change depending on the number of sides of the polygon. If there is only 1 subpath and the complexity is set to 2, then you will always create a simple star. For more complex stars, we recommend you experiment with the recorder to determine the appropriate values.
6	.**Star** Set to TRUE (-1) to enable the StarComplexity parameter. Set to FALSE (0) to ignore this parameter. This parameter must be TRUE if you want to create a star-shaped polygon.
7	.**StarComplexity** Lets you specify the distance that the start node is from the center of the shape. If this parameter is set to 0, the start node will remain at the outer edge of the shape. The closer this parameter's value is to the .lMaxComplexity value, the closer the start node is to the center of the shape. This parameter is equivalent to the Sharpness slider for polygons (as opposed to stars).
8	.**MaxComplexity** Lets you specify the maximum value for star complexity.

{button ,AL(`CLS_CorelScript;FNC_CreateSymPolygon')} **Related Topics**

CorelScript.RecorderApplyPerspective

Function **RecorderApplyPerspective**(ByVal **Type** AS Long, ByVal **Flags** AS Long, ByVal **Box0X** AS Long, ByVal **Box0Y** AS Long, ByVal **Box1X** AS Long, ByVal **Box1Y** AS Long, ByVal **Box2X** AS Long, ByVal **Box2Y** AS Long, ByVal **Box3X** AS Long, ByVal **Box3Y** AS Long, ByVal **VPHorizRef** AS Long, ByVal **VPHorizX** AS Long, ByVal **VPHorizY** AS Long, ByVal **VPVertRef** AS Long, ByVal **VPVertX** AS Long, ByVal **VPVertY** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderApplyPerspective')} **Related Topics**

CorelScript.RegisterObject

Function **RegisterObject**(ByVal **ObjectID** AS String) AS Boolean

0 CorelScript

0 Member of **CorelScript**

Parameters	**Description**
ObjectID	Description of 'ObjectID' goes here

Return value

0 Returns Boolean

1 Example

2 Example of usage goes here

{button ,AL(`CLS_CorelScript;FNC_RegisterObject')} **Related Topics**

CorelScript.UnRegisterObject

Function **UnRegisterObject**(ByVal **ObjectID** AS String) AS Boolean

0 CorelScript

0 Member of **CorelScript**

Parameters	Description
ObjectID	Description of 'ObjectID' goes here

Return value

0 Returns Boolean

1 Example

2 Example of usage goes here

{button ,AL(`CLS_CorelScript;FNC_UnRegisterObject')} **Related Topics**

CorelScript.RecorderObjectScaleInfo

Function **RecorderObjectScaleInfo**(ByVal **ScaledSizeX** AS Long, ByVal **ScaledSizeY** AS Long, ByVal **DisplacementX** AS Long, ByVal **DisplacementY** AS Long) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_RecorderObjectScaleInfo')} **Related Topics**

CorelScript.AppendCurveLine

Function **AppendCurveLine**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long) AS Long

0 CorelScript

Description

This command adds a line segment to an existing curve. An open curve must be selected.

Parameter	Description
.**X1**	Lets you specify the X-coordinate of the start point in tenths of a micron, relative to the center of the page.
0	.**Y1** Lets you specify the Y-coordinate of the start point in tenths of a micron, relative to the center of the page.
1	.**X2** Lets you specify the X-coordinate of the end point in tenths of a micron, relative to the center of the page.
2	.**Y2** Lets you specify the Y-coordinate of the end point in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_AppendCurveLine')} **Related Topics**

CorelScript.BeginDrawFreehand

Function **BeginDrawFreehand**(ByVal **ConvertToDPCoords** AS Boolean, ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This commandcreates the first point of a freehand segment. Additional points are added to the segment by using the .AddFreehandPoint command. The curve itself is not added until the .EndDrawFreehand command is called.

Parameter	Description
.**ConvertToDPCoords**	Set to TRUE (-1) to convert the X and Y coordinates to physical coordinates on the screen. This parameter should always be set to true unless you know that the coordinates you are using are already physical coodinates on the screen.
0	.**X** Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Y** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_BeginDrawFreehand')} **Related Topics**

CorelScript.AddFreehandPoint

Function **AddFreehandPoint**(ByVal **ConvertToDPCoords** AS Boolean, ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This command adds a point to a freehand curve created by using the .BeginDrawFreehand command. The segment itself is not added until .EndDrawFreehand command is called.

Parameter	Description
.**ConvertToDPCoords**	Set to TRUE (-1) to convert the X and Y coordinates to physical coordinates on the screen. This parameter should always be set to true unless you know that the coordinates you are using are already physical coodinates on the screen.
0	.**X** Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Y** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_AddFreehandPoint')} **Related Topics**

CorelScript.EndDrawFreehand

Function **EndDrawFreehand**(ByVal **StraightTightness** AS Long, ByVal **CurveTightness** AS Long, ByVal **CornerThreshold** AS Long, ByVal **SnapTightness** AS Long) AS Long

0 CorelScript

Description

This command ends a set of frehand drawing commands that began with the .BeginDrawCurve command.

Parameter	Description
.**StraightTightness**	Sets the amount the curve can vary from a straight path and still be treated as straight. The higher the value, the less accurate the line needs to be. The valid range is from 1 to 10 pixels.
0	.**CurveTightness** Determines how closely the curve will match the position of each point. The lower the number, the more accurate the match. The valid range is from 1 to 10 pixels.
1	.**CornerThreshold** Sets the limit at which each corner node is cusped (as opposed to smooth). A node is more likely to be cusped if the value is lower. The valid range is from 1 to 10 pixels.
2	.**SnapTightness** Determines how close two end nodes must be to join automatically. The valid range is from 1 to 10 pixels.

{button ,AL(`CLS_CorelScript;FNC_EndDrawFreehand')} **Related Topics**

CorelScript.InitBezierTool

Function **InitBezierTool**() AS Long

0 CorelScript

Description

This command must preceed the .BeginDrawBezier command.

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`CLS_CorelScript;FNC_InitBezierTool')} **Related Topics**

CorelScript.BeginDrawBezier

Function **BeginDrawBezier**(ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **Cusp** AS Boolean) AS Long

0 CorelScript

Description

This command creates the first point of a bezier segment. The second point is added by the .AddBezierPoint command. The segment itself is not added until the .EndDrawBezier command is called. This command must be preceeded by the .InitBezierTool command.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Cusp** Set to TRUE (-1) to make the new node cusped. Set to FALSE (0) to make it symmetrical (except for end nodes).

Note

- The bezier commands include:
  .AddBezierPoint
  .MoveBezierControl

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`CLS_CorelScript;FNC_BeginDrawBezier')} **Related Topics**

CorelScript.AddBezierPoint

Function **AddBezierPoint**(ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **Constrain** AS Boolean, ByVal **Cusp** AS Boolean) AS Long

0 CorelScript

Description

This command creates the second point of a bezier segment created by using the .BeginDrawBezier command. The segment itself is not added until the .EndDrawBezier command is called.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Constrain** Set to TRUE (-1) to use the constrain angle when positioning the point.
2	.**Cusp** Set to TRUE (-1) to make the new node cusped. Set to FALSE (0) to make it symmetrical (except for end nodes).

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`CLS_CorelScript;FNC_AddBezierPoint')} **Related Topics**

CorelScript.MoveBezierControl

Function **MoveBezierControl**(ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **Constrain** AS Boolean) AS Long

0 CorelScript

Description

This command controls the position of bezier node control points created with the .BeginDrawBezier and .AddBezierPoint commands.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the control point's position in tenths of a micron, relative to the center of the page.
0	.**Y** Lets you specify the Y-coordinate of the control point's position in tenths of a micron, relative to the center of the page.
1	.**Constrain** Set to TRUE (-1) to use the constrain angle when positioning the control points.

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`CLS_CorelScript;FNC_MoveBezierControl')} **Related Topics**

CorelScript.EndDrawBezier

Function **EndDrawBezier**() AS Long

0 CorelScript

Description

This command ends a set of bezier creation commands that began with the .BeginDrawBezier command.

Note

- The bezier commands include:
  .AddBezierPoint
  .MoveBezierControl

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`CLS_CorelScript;FNC_EndDrawBezier')} **Related Topics**

CorelScript.ShareExtrudeVP

Function **ShareExtrudeVP**(ByVal **ExtrudeIndex** AS Long, ByVal **VPToShareIndex** AS Long, ByVal **SharedVP** AS Boolean) AS Long

0 CorelScript

Description

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`CLS_CorelScript;FNC_ShareExtrudeVP')} **Related Topics**

CorelScript.FileExit

Function **FileExit**(ByVal **PromptUser** AS Boolean) AS Boolean

0 CorelScript

Description

This command exits the application.

Return Value

Returns one of the following values:

- TRUE (-1) the application was closed
- FALSE (0) the application was not closed.

Parameter	Description
.**PromptUser**	Set to TRUE (-1) to prompt the user before closing the application. Set to FALSE (0) to exit the application without prompting the user.

Note

- This command must be preceded by the .FileSave command or changes will be lost.

{button ,AL(`CLS_CorelScript;FNC_FileExit')} **Related Topics**

CorelScript.BeginDrawArrow

Function **BeginDrawArrow**(ByVal **LeftArrow** AS Boolean, ByVal **LineOffset** AS Long, ByVal **NumOfPoints** AS Long) AS Long

0 CorelScript

Description

This command initializes a block of arrowhead creation commands. This block must include one or more instances of the .AddArrowPoint command and this block must end with the .EndDrawArrow command. To add the arrowhead to a path, use the .SetOutlineArrow command.

Parameter	Description
.**LeftArrow**	Set to TRUE (-1) to create a left-facing arrowhead. Set to FALSE (0) to create a right-facing arrowhead.
0	.**LineOffset** Lets you specify the distance between the end of the path and the arrowhead.
1	.**NumOfPoints** Lets you specify the number of nodes in your arrowhead.

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`CLS_CorelScript;FNC_BeginDrawArrow')} **Related Topics**

CorelScript.AddArrowPoint

Function **AddArrowPoint**(ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **Enabled** AS Boolean, ByVal **Letter** AS Boolean, ByVal **User** AS Boolean, ByVal **Closed** AS Boolean, ByVal **Continuity** AS Long, ByVal **NodeType** AS Long) AS Long

0 CorelScript

Description

This command must be part of a block of commands that begins with the .BeginDrawArrow command and ends with the .EndDrawArrow command. To add the arrowhead to a path, use the .SetOutlineArrow command.

Parameter	Description
.**X**	Lets you specify the X-coordinate of the node in tenths of a micron, relative to the center of the arrowhead area.
0	.**Y** Lets you specify the Y-coordinate of the node in tenths of a micron, relative to the center of the arrowhead area.
1	.**Enabled** Always set to FALSE (0)
2	.**Letter** Always set to FALSE (0)
3	.**User** Set to TRUE (-1) to make this node selectable.
4	.**Closed** Set to TRUE (-1) to make the arrowhead closed.
5	.**Continuity** 0 = cusp node 1 = smooth node 2 = symmetrical node
6	.**NodeType** 1 = straight line segment 2 = curved line segment

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`CLS_CorelScript;FNC_AddArrowPoint')} **Related Topics**

CorelScript.EndDrawArrow

Function **EndDrawArrow**() AS Long

0 CorelScript

Description

This command ends a block of arrowhead creation commands. This block must include one or more instances of the .AddArrowPoint command and this block must begin with the .BeginDrawArrow command.

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`CLS_CorelScript;FNC_EndDrawArrow')} **Related Topics**

CorelScript.SetOutlineWidth

Function **SetOutlineWidth**(ByVal **Width** AS Long) AS Long

0 CorelScript

Description

This command changes the width of the outline of the selected object.

Parameter	Description
.**Width**	Lets you specify the width of the outline.

{button ,AL(`CLS_CorelScript;FNC_SetOutlineWidth')} **Related Topics**

CorelScript.SetOutlineMiscProperties

Function **SetOutlineMiscProperties**(ByVal **Type** AS Long, ByVal **Style** AS Long, ByVal **Corners** AS Long, ByVal **LineCaps** AS Long, ByVal **Aspect** AS Long, ByVal **Angle** AS Long, ByVal **BehindFill** AS Boolean, ByVal **ScalePen** AS Boolean) AS Long

0 CorelScript

Description

This command sets the outline properties of the selected object.

Parameter	Description
.**Type**	Lets you specify the outline type: 0 = None 1 = Solid 2 = Dot - Dash
0	.**Style** Calls .DotDash from the ApplyOutline command.
1	.**Corners** 0 = Mitered
2	1 = Beveled
3	2 = Rounded
4	.**LineCaps** 0 = Square
5	1 = Rounded
6	2 = Extended Square
7	.**Aspect** Lets you specify the stretch field which adjusts the width of the nib. Valid values range from 1 to 100 percent.
8	.**Angle** Lets you specify the angle of the nib's edge, in tenths of degrees.
9	.**BehindFill** Set to TRUE (-1) to position the outline behind the fill. Set to FALSE (0) to position the outline in front of the fill.
10	.**ScalePen** Set to TRUE (-1) to scale the outline when the object is scaled.

{button ,AL(`CLS_CorelScript;FNC_SetOutlineMiscProperties')} **Related Topics**

CorelScript.SetOutlineArrow

Function **SetOutlineArrow**(ByVal **ArrowType** AS Long) AS Long

0 CorelScript

Description

This command changes the arrowhead of the outline of the selected object. This command follows a block of arrowhead creation commands (see example).

Parameter	Description
.**ArrowType**	Lets you specify the arrow position. 0 0 = left arrow 1 = right arrow 2 = both arrows

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`CLS_CorelScript;FNC_SetOutlineArrow')} **Related Topics**

CorelScript.SetPageOrientation

Function **SetPageOrientation**(ByVal **Orient** AS Long) AS Long

0 CorelScript

Description

This command changes the orientation of the page.

Parameter	Description
.**Orient**	DRAW_ORIENT_PORTRAIT = portrait DRAW_ORIENT_LANDSCAPE = landscape

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetPageOrientation command.

{button ,AL(`CLS_CorelScript;FNC_SetPageOrientation')} **Related Topics**

CorelScript.ConvertColor

Function **ConvertColor**(ByVal **ToColorModel** AS Long, ByRef **ToV1** AS Long, ByRef **ToV2** AS Long, ByRef **ToV3** AS Long, ByRef **ToV4** AS Long, ByRef **ToV5** AS Long, ByRef **ToV6** AS Long, ByRef **ToDensity** AS Long) AS Long

0 CorelScript

Description

This command converts one color model to another color model.

Parameter	Description
.**ToColorModel**	Lets you specify the destination folder.
0	.**V1** Returns the first color component.
1	.**V2** Returns the second color component.
2	.**V3** Returns the third color component.
3	.**V4** Returns the fourth color component.
4	.**V5** Returns the fifth color component.
5	.**V6** Returns the sixth color component.
6	.**V7** Returns the color density from 0 to 100

Example

```
.GetUniformFillColor InModel, In1, In2, In3, In4, In5, In6, In7
.StoreColor InModel, In1, In2, In3, In4, In5, In6, In7
.ConvertColor DRAW_COLORMODEL_RGB, RedVal, GREENVAL, BLUEVAL, 0, 0, 0, 0
```

{button ,AL(`CLS_CorelScript;FNC_ConvertColor')} **Related Topics**

CorelScript.GetFountainFillColor

Function **GetFountainFillColor**(ByVal **Index** AS Long, ByRef **Position** AS Long, ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef **V2** AS Long, ByRef **V3** AS Long, ByRef **V4** AS Long, ByRef **V5** AS Long, ByRef **V6** AS Long, ByRef **Density** AS Long) AS Long

0 CorelScript

Description

This command Retrieves a color from a fountain fill.

Parameter	Description
.**Index**	The index number of the color you want to get. Use the .GetFountainFill command to find out how many colors are in a fountain fill. Valid index numbers will range from 0 to the number of colors minus 1. If you use the value 100, you will always get the end color.
0	.**Position** Return the position of the color within the fill.
1	.**ColorModel** Returns the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome
2	.**V1** Returns the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
3	.**V2** Returns the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
4	.**V3** Returns the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
5	.**V4** Returns the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
6	.**V5** Returns the fifth color component for .plColorModel.
7	.**V6** Returns the sixth color component for .plColorModel.
8	.**Density** Returns the color density.

{button ,AL(`CLS_CorelScript;FNC_GetFountainFillColor')} **Related Topics**

CorelScript.BeginEditObject

Function **BeginEditObject**() AS Long

0 CorelScript

Description

This command initializes a block of node editing commands that includes one or more instances of the .EditObjectCommand and ends with the .EndEditObject command.

{button ,AL(`CLS_CorelScript;FNC_BeginEditObject')} **Related Topics**

CorelScript.EditObjectCommand

Function **EditObjectCommand**(ByVal **Cmd** AS Long, ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **Key** AS Long, ByVal **AddToSelection** AS Boolean) AS Long

0 CorelScript

Description

This command lets you change the shape of objects by editing their nodes. This command must be part of a block of commands that begins with the .BeginEditObject command and ends with the .EndEditObject command.

Parameter	Description
.**Cmd**	0 = This value is equivalent to releasing the mouse button. Use with the .lX parameter and the .lY parameter. 1 = This value is equivalent to pressing one of the nudge or super-nudge keys. Use the .lKey parameter to indicate which nudge key is used. 2 = This value is equivalent to pressing a key on the keyboard. Use the .lKey parameter to indicate which key is used. 3 = This value is equivalent to pressing down the mouse button. Use with the .lX parameter and the .lY parameter. 4 = This value is equivalent to the add node option. 5 = This value is equivalent to the delete node option. 6 = This value is equivalent to the join nodes option. 7 = This value is equivalent to the break path option. 8 = This value makes a node cusped. 9 = This value makes a node smooth. 10 = This value makes a segment straight. 11 = This value makes a segment curved. 12 = This value makes a node symmetrical. 13 = This value is equivalent to the auto-reduce nodes option. 14 = This value is equivalent to the extract subpath option. 16 = This value toggles the elastic option.
0	.**X** Lets you specify the X-coordinate of a node in tenths of a micron, relative to the center of the page.
1	.**Y** Lets you specify the Y-coordinate of a node in tenths of a micron, relative to the center of the page.
2	.**Key** If .lCmd = 1 3 0 = nudge left 1 = nudge right 2 = nudge up 3 = nudge down 4 = super-nudge left 5 = super-nudge right 6 = super-nudge up 7 = super-nudge down 4 If .lCmd = 2 5 0 = HOME 1 = END 2 = TAB 3 = ESC
6	.**AddToSelection** Set to TRUE (-1) to select multiple nodes (equivalent to pressing SHIFT).

{button ,AL(`CLS_CorelScript;FNC_EditObjectCommand')} **Related Topics**

CorelScript.EndEditObject

Function **EndEditObject**() AS Long

0 CorelScript

Description

This command ends a block of node editing commands that includes one or more instances of the .EditObjectCommand and begins with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_EndEditObject')} **Related Topics**

CorelScript.CloseCurve

Function **CloseCurve**() AS Long

0 CorelScript

Description

This command closes the selected open path.

{button ,AL(`CLS_CorelScript;FNC_CloseCurve')} **Related Topics**

CorelScript.GetGuidelineInformation

Function **GetGuidelineInformation**(ByVal **Index** AS Long, ByRef **Point1X** AS Long, ByRef **Point1Y** AS Long, ByRef **Point2X** AS Long, ByRef **Point2Y** AS Long, ByRef **InterceptX** AS Long, ByRef **InterceptY** AS Long, ByRef **Angle** AS Long, ByRef **Locked** AS Boolean) AS Long

0 CorelScript

Description

This command retrieves all the information available about a specific guideline. The .plPoint parameters are equivalent to the coordinates used in the .CreateGuidlineUsingTwoPoints command, and the .plIntercept and .plAngle parameters are equivalent to the parameters used in the .CreateGuidelineUsingAngle command.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
0	.**Point1X** Returns the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
1	.**Point1Y** Returns the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
2	.**Point2X** Returns the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
3	.**Point2Y** Returns the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
4	.**InterceptX** Returns the X-coordinate of the point that relates to .plAngle in tenths of a micron, relative to the center of the page.
5	.**InterceptY** Returns the Y-coordinate of the point that relates to .plAngle in tenths of a micron, relative to the center of the page.
6	.**Angle** Returns the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).
7	.**Locked** Returns TRUE (-1) if guidelines are locked. Returns FALSE (0) if the guidelines are unlocked.

{button ,AL(`CLS_CorelScript;FNC_GetGuidelineInformation')} **Related Topics**

CorelScript.GetNumberOfGuidelines

Function **GetNumberOfGuidelines**() AS Long

0 CorelScript

Description

This function return the number of guidelines. You can use this function to determine the index of a guidline. The range of possible guideline index values is 0 to the number of guidelines minus 1.

Return Value

Returns the following value:

- the number of guidelines

{button ,AL(`CLS_CorelScript;FNC_GetNumberOfGuidelines')} **Related Topics**

CorelScript.MoveGuidelineUsingTwoPointsByIndex

Function **MoveGuidelineUsingTwoPointsByIndex**(ByVal **Index** AS Long, ByVal **Point1X** AS Long, ByVal **Point1Y** AS Long, ByVal **Point2X** AS Long, ByVal **Point2Y** AS Long, ByVal **Locked** AS Boolean) AS Long

0 CorelScript

Description

This command moves a specific guideline to a location using two sets of coordinates to place the guidleine.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
0	.**Point1X** Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
1	.**Point1Y** Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
2	.**Point2X** Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
3	.**Point2Y** Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
4	.**Locked** Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`CLS_CorelScript;FNC_MoveGuidelineUsingTwoPointsByIndex')} **Related Topics**

CorelScript.MoveGuidelineUsingAngleByIndex

Function **MoveGuidelineUsingAngleByIndex**(ByVal **Index** AS Long, ByVal **InterceptX** AS Long, ByVal **InterceptY** AS Long, ByVal **Angle** AS Long, ByVal **Locked** AS Boolean) AS Long

0 CorelScript

Description

This command moves a specific guideline to a location using an intersection point and an angle to specify where to put the new guideline.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
0	.**InterseptX** Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**InterseptY** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
2	.**Angle** Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).
3	.**Locked** Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`CLS_CorelScript;FNC_MoveGuidelineUsingAngleByIndex')} **Related Topics**

CorelScript.LockGuidelineByIndex

Function **LockGuidelineByIndex**(ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This command locks a specific guideline.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`CLS_CorelScript;FNC_LockGuidelineByIndex')} **Related Topics**

CorelScript.UnlockGuidelineByIndex

Function **UnlockGuidelineByIndex**(ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This command unlocks a specific guideline.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`CLS_CorelScript;FNC_UnlockGuidelineByIndex')} **Related Topics**

CorelScript.DeleteGuidelineByIndex

Function **DeleteGuidelineByIndex**(ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This command deletes a specific guideline.

Parameter	Description
.**Index**	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`CLS_CorelScript;FNC_DeleteGuidelineByIndex')} **Related Topics**

CorelScript.DeleteGuidelineUsingTwoPoints

Function **DeleteGuidelineUsingTwoPoints**(ByVal **Point1X** AS Long, ByVal **Point1Y** AS Long, ByVal **Point2X** AS Long, ByVal **Point2Y** AS Long) AS Boolean

0 CorelScript

Description

This command deletes a specific guideline using two sets of coordinates.

Parameter	Description
.**Point1X**	Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
0	.**Point1Y** Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
1	.**Point2X** Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
2	.**Point2Y** Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.

{button ,AL(`CLS_CorelScript;FNC_DeleteGuidelineUsingTwoPoints')} **Related Topics**

CorelScript.DeleteGuidelineUsingAngle

Function **DeleteGuidelineUsingAngle**(ByVal **IntersectX** AS Long, ByVal **IntersectY** AS Long, ByVal **Angle** AS Long) AS Boolean

0 CorelScript

Description

This command deletes a specific guideline based on a set of coordinates and an angle.

Parameter	Description
.**IntersectPointX**	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
0	.**IntersectPointY** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Angle** Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).

{button ,AL(`CLS_CorelScript;FNC_DeleteGuidelineUsingAngle')} **Related Topics**

CorelScript.CreateGuidelineUsingAngle

Function **CreateGuidelineUsingAngle**(ByVal **IntersectPointX** AS Long, ByVal **IntersectPointY** AS Long, ByVal **Angle** AS Long, ByVal **Locked** AS Boolean) AS Long

0 CorelScript

Description

This command deletes a specific guideline based on a set of coordinates and an angle.

Parameter	Description
.**IntersectPointX**	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
0	.**IntersectPointY** Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
1	.**Angle** Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).

{button ,AL(`CLS_CorelScript;FNC_CreateGuidelineUsingAngle')} **Related Topics**

CorelScript.CreateGuidelineUsingTwoPoints

Function **CreateGuidelineUsingTwoPoints**(ByVal **Point1X** AS Long, ByVal **Point1Y** AS Long, ByVal **Point2X** AS Long, ByVal **Point2Y** AS Long, ByVal **Locked** AS Boolean) AS Long

0 CorelScript

Description

This command creates a guideline at a specific location using two sets of coordinates to place the guidleine.

Parameter	Description
.**Point1X**	Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
0 .**Point1Y**	Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
1 .**Point2X**	Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
2 .**Point2Y**	Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
3 .**Locked**	Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`CLS_CorelScript;FNC_CreateGuidelineUsingTwoPoints')} **Related Topics**

CorelScript.RemoveAllGuidelines

Function **RemoveAllGuidelines**() AS Long

0 CorelScript

Description

This command removes all guidelines.

{button ,AL(`CLS_CorelScript;FNC_RemoveAllGuidelines')} **Related Topics**

CorelScript.GetPageSize

Function **GetPageSize**(ByRef **Width** AS Long, ByRef **Height** AS Long) AS Long

0 CorelScript

Description

This command returns the width and height of the document page.

Parameter	Description
.**Width**	Returns the width of the page in tenths of a micron.
0 .**Height**	Returns the height of the page in tenths of a micron.

{button ,AL(`CLS_CorelScript;FNC_GetPageSize')} **Related Topics**

CorelScript.GetDocumentCount

Function **GetDocumentCount**() AS Long

0 CorelScript

Description

This command returns the number of open documents.

Example

```
ICount& = .GetDocumentCount()
```

{button ,AL(`CLS_CorelScript;FNC_GetDocumentCount')} **Related Topics**

CorelScript.GetDocumentName

Function **GetDocumentName**() AS String

0 CorelScript

Description

This command returns the name of the current document.

{button ,AL(`CLS_CorelScript;FNC_GetDocumentName')} **Related Topics**

CorelScript.SetDocVisible

Function **SetDocVisible**(ByVal **Show** AS Boolean) AS Long

0 CorelScript

Description

This command makes the current document visible or hidden.

Parameter	Description
.**Show**	Set to TRUE (-1) to make a document visible. Set to FALSE (0) to hide a document.

{button ,AL(`CLS_CorelScript;FNC_SetDocVisible')} **Related Topics**

CorelScript.SetCurrentDocument

Function **SetCurrentDocument**() AS Boolean

0 CorelScript

Description

This command makes the selected document the current document.

{button ,AL(`CLS_CorelScript;FNC_SetCurrentDocument')} **Related Topics**

CorelScript.CreateDimension

Function **CreateDimension**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **X3** AS Long, ByVal **Y3** AS Long, ByVal **Style** AS Long, ByVal **WitnessExtension** AS Long, ByVal **WitnessGap** AS Long, ByVal **LabelGap** AS Long) AS Long

0 CorelScript

Description

This command creates a horizontal, vertical or slanted dimension line.

Parameter	Description
.X1, Y1	Lets you specify the start point coordinate of the dimension line.
0	**.X2, Y2** Lets you specify the end point coordinate of the dimension line.
1	**.X3, Y3** Lets you specify the dimension text location.
2	**.Style** 0=vertical dimension 3 1=horizontal dimension 4 2=slanted dimension
5	**.WitnessExtention** Lets you specify the length of the extension line that protrudes beyond the dimension line.
6	**.WitnessGap** Lets you specify the distance between the snapped to object and the extension line.
7	**.LabelGap** Lets you specify the amount of space between the text and the dimension line.

{button ,AL(`CLS_CorelScript;FNC_CreateDimension')} **Related Topics**

CorelScript.CreateConnector

Function **CreateConnector**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **Placement** AS Boolean) AS Long

0 CorelScript

Description

This command connects objects with a line. When you move an object that has a connector line attached the connetor line also moves.

Parameter	Description
.X1, Y1	Lets you specify the first coordinate of the line.
0	**.X2, Y2** Lets you specify the second coordinate of the line.
1	**.Placement** Set to TRUE to keep connector line fixed to the nodes that it was originally attached to. Set to FALSE to draw the shortest line between the two objects it connects.

{button ,AL(`CLS_CorelScript;FNC_CreateConnector')} **Related Topics**

CorelScript.CreateCallout

Function **CreateCallout**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **X3** AS Long, ByVal **Y3** AS Long, ByVal **Text** AS String) AS Long

0 CorelScript

Description

This command is used to create a callout line that points to and labels an object.

Parameter	Description
.X1, Y1	Lets you specify where the first callout segment starts.
0	**.X2, Y2** Lets you specify where the first callout segment ends.
1	**.X3, Y3** Lets you specify the location of the callout text.
2	**.Text** Lets you specify the callout text.

{button ,AL(`CLS_CorelScript;FNC_CreateCallout')} **Related Topics**

CorelScript.CreateAngleDimension

Function **CreateAngleDimension**(ByVal **x1** AS Long, ByVal **y1** AS Long, ByVal **x2** AS Long, ByVal **y2** AS Long, ByVal **X3** AS Long, ByVal **Y3** AS Long, ByVal **X4** AS Long, ByVal **Y4** AS Long, ByVal **LargeAngle** AS Boolean, ByVal **WitnessExtension** AS Long, ByVal **WitnessGap** AS Long, ByVal **LabelGap** AS Long) AS Long

0 CorelScript

Description

This command is used to create an angular dimension.

Parameter	Description
.1X1, 1Y1	Lets you specify the apex coordinates.
0	**.1X2, 1Y2** Lets you specify the baseline coordinates.
1	**.1X3, 1Y3** Lets you specify the endline coordinates.
2	**.1X4, 1Y4** Lets you specify the text point.
3	.**LargeAngle** Set to TRUE the angle measured is <= 180 degrees. Set to FALSE the angle measured is >180 degrees.
4	.**WitnessExtension** Lets you specify the length of the extension line that protrudes beyond the dimension line.
5	.**WitnessGap** Lets you specify the distance between the snapped to object and the extension line.
6	.**LabelGap** Lets you specify the amount of space between the text and the dimension line.

{button ,AL(`CLS_CorelScript;FNC_CreateAngleDimension')} **Related Topics**

CorelScript.EditDimensionLabel

Function **EditDimensionLabel**(ByVal **Placement** AS Integer, ByVal **Horizontal** AS Boolean, ByVal **Center** AS Boolean, ByVal **Dynamic** AS Boolean, ByVal **Style** AS Integer, ByVal **Precision** AS Integer, ByVal **Units** AS Integer, ByVal **ShowUnits** AS Boolean, ByVal **Prefix** AS String, ByVal **Suffix** AS String) AS Long

0 CorelScript

Description

This command lets you change the properties of a vertical, horizontal or slanted dimension label.

Parameter	Description
.**Placement**	0=Places text above the dimension line.
	0 1=Places text within the dimension line.
	1 2=Places text below the dimension line.
2	.**Horizontal** Set to TRUE places the text horizontally.
3	.**Center** Set to TRUE centers the text in the dimension line.
4	.**Dynamic** Set to TRUE the dimension text changes automatically when the objects is stretched or skewed.
5	.**Style** 0=Decimal
	6 1=Fractional
	7 2=U.S. Engineering
	8 3=U.S. Architectural
9	.**Precision** 0=0 or 0
	10 1=0.0 or 01/2 or 0'-0" or 0'-0" depending on .sStyle
	11 2=0.00 or 01/4 or 0'-0.0" or 0'-01/4" depending on .sStyle
	12 3=0.000 or 01/8 or 0'-0.00" or 0'-01/8" depending on .sStyle
	13 4=0.0000 or 01/16 or 0'-0.000" or 0'-01/16" depending on .sStyle
	14 5=0.00000 or 01/32 or 0'-0.0000" or 0'-01/32" depending on .sStyle
	15 6=0.000000 or 01/64 or 0'-0.00000" or 0'-01/64" depending on .sStyle
	16 7=0.0000000 or 01/128 or 0'-0.000000" or 0'-01/128" depending on .sStyle
	17 8=0.00000000 or 01/256 or 0'-0.0000000" or 0'-01/256" depending on .sStyle
	18 9=0.000000000 or 01/512 or 0'-0.00000000" or 0'-01/512" depending on .sStyle
	19 10=0.0000000000 or 01/1024 or 0'-0.000000000" or 0'-01/1024" depending on .sStyle
20	.**Units** 0="
	21 1=in
	22 2=inches
	23 3='
	24 4=ft
	25 5=mi
	26 6=miles
	27 7=yds
	28 8=yards
	29 9=m
	30 10=meters
	31 11=km
	32 12=kilometers
	33 13=cm
	34 14=centimeters
	35 15=mm
	36 16=millimeters
	37 17=picas
	38 18=points
	39 19=ciceros
	40 20=didots
41	.**ShowUnits** Set to TRUE displays the units beside the dimension text.
42	.**Prefix** Lets you specify that the prefix is attached to the dimension text.
43	.**Suffix** Lets you specify that the suffix is attached to the dimension text.

{button ,AL(`CLS_CorelScript;FNC_EditDimensionLabel')} **Related Topics**

CorelScript.EditAngleDimensionLabel

Function **EditAngleDimensionLabel**(ByVal **Dynamic** AS Boolean, ByVal **Precision** AS Integer, ByVal **Units** AS Integer, ByVal **ShowUnits** AS Boolean, ByVal **Prefix** AS String, ByVal **Suffix** AS String) AS Long

0 CorelScript

Description

This command lets you change the properties of an angle dimension label.

Parameter	Description
.**Dynamic**	Set to TRUE the dimension text changes automatically when the objects is stretched or skewed.
0	.**Precision** 0=0
	1 1=0.0
	2 2=0.00
	3 3=0.000
	4 4=0.0000
	5 5=0.00000
	6 6=0.000000
	7 7=0.0000000
	8 8=0.00000000
	9 9=0.000000000
	10 10=0.0000000000
11	.**Units** 0=degrees
	12 1=radians
	13 2=gradians
14	.**ShowUnits** Set to TRUE displays the units beside the dimension text.
15	.**Prefix** Lets you specify that the prefix is attached to the dimension text.
16	.**Suffix** Lets you specify that the suffix is attached to the dimension text.

{button ,AL(`CLS_CorelScript;FNC_EditAngleDimensionLabel')} **Related Topics**

CorelScript.ApplyDistortion

Function **ApplyDistortion**(ByVal **New** AS Boolean, ByVal **DistortionType** AS Long, ByVal **Amplitude** AS Long, ByVal **Frequency** AS Long, ByVal **Seed** AS Long, ByVal **Angle** AS Long, ByVal **CenterX** AS Long, ByVal **CenterY** AS Long, ByVal **Flag** AS Long) AS Long

0 CorelScript

Description

This command applies a contour to the selected object.

Parameter	Description
.**New**	Set to TRUE will add to existing distortion. Set to FALSE clears the existing distortion.
0	.**DistortionType** 1=Push & Pull
1	2=Zipper
2	3=Twister
3	.**Amplitude** 1=-200-200
4	2=0-100
5	3=N/A
6	.**Frequency** 1=N/A
7	2=Peaks/Curve
8	3=N/A
9	.**Seed** If the .Random flag is set then .Seed sets the number of seeds for the zipper distortion.
10	.**Angle** 1=N/A
11	2=N/A
12	3=the maximum twist angle
13	.**CenterX** Lets you specify the offset in X from the center of the distortion.
14	.**CenterY** Lets you specify the offset in Y from the center of the distortion.
15	.**Flag** 1=the smoothness of the zipper
16	2=the random amplitude of the zipper
17	4=set the acceleration of the distortion towards the center of the zipper.

{button ,AL(`CLS_CorelScript;FNC_ApplyDistortion')} **Related Topics**

CorelScript.SetCornerRoundness

Function **SetCornerRoundness**(ByVal **Roundness** AS Long, ByVal **Roundness2** AS Long, ByVal **Roundness3** AS Long, ByVal **Roundness4** AS Long) AS Long

0 CorelScript

Description

This command sets the corner roundness of the selected object.

Parameter	Description
.**Roundness**	Lets you specify the roundness of the corners in tenths of a percent.
0	.**Roundness2** Lets you specify the roundness of the corners in tenths of a percent.
1	.**Roundness3** Lets you specify the roundness of the corners in tenths of a percent.
2	.**Roundness4** Lets you specify the roundness of the corners in tenths of a percent.

{button ,AL(`CLS_CorelScript;FNC_SetCornerRoundness')} **Related Topics**

CorelScript.SetEllipseProperties

Function **SetEllipseProperties**(ByVal **Arc** AS Boolean, ByVal **StartAngle** AS Long, ByVal **EndAngle** AS Long) AS Long

0 CorelScript

Description

This command sets the ellipse properties.

Parameter	Description
.**Arc**	Set to TRUE = Arc. Set to FALSE = Ellipse/Pie
0	.**StartAngle** Lets you specify the start angle in millionths of a degree.
1	.**EndAngle** Lets you specify the end angle in millionths of a degree.

{button ,AL(`CLS_CorelScript;FNC_SetEllipseProperties')} **Related Topics**

CorelScript.SetPolygonProperties

Function **SetPolygonProperties**(ByVal **Star** AS Boolean, ByVal **Points** AS Long, ByVal **Sharpness** AS Long) AS Long

0 CorelScript

Description

This command sets the polygon properties of the selected object.

Parameter	Description
.**Star**	If set to TRUE = Star. If set to FALSE = Polygon.
0	.**Points** Lets you specify the number of points/edges in a polygon.
1	.**Sharpness** Speficies the sharpness of the star.

{button ,AL(`CLS_CorelScript;FNC_SetPolygonProperties')} **Related Topics**

CorelScript.StoreColor

Function **StoreColor**(ByVal **ColorModel** AS Long, ByVal **V1** AS Long, ByVal **V2** AS Long, ByVal **V3** AS Long, ByVal **V4** AS Long, ByVal **V5** AS Long, ByVal **V6** AS Long, ByVal **V7** AS Long, ByVal **Position** AS Long) AS Long

0 CorelScript

Description

This command sets the color that will be applied to the selected object.

Parameter	Description
.**ColorModel**	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 0 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
1	.**Color1** Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
2	.**Color2** Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
3	.**Color3** Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
4	.**Color4** Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

Example

```
.StoreColor DRAW_COLORMODEL_CMYK255, 0, 255, 0, 0, 0, 0, 0
.SetOutlineColor
```

The above example stores the Magenta color for use in the SetOutlineColor command.

{button ,AL(`CLS_CorelScript;FNC_StoreColor')} **Related Topics**

CorelScript.ApplyDropShadow

Function **ApplyDropShadow**(ByVal **HorizOffset** AS Long, ByVal **VertOffset** AS Long, ByVal **Opacity** AS Long, ByVal **Feather** AS Long, ByVal **FeatherType** AS Long, ByVal **FeatherEdge** AS Long, ByVal **PerspectiveType** AS Long, ByVal **PerspectiveSkewAngle** AS Long, ByVal **PerspectiveStretch** AS Double, ByVal **Fade** AS Long, ByVal **IdenticalValues** AS Boolean) AS Long

0 CorelScript

Description

This command applies a drop shadow to the selected object.

Parameter	Description
.**HorizOffset**	Lets you specify the horizontal offset in tenths of a micron.
0	.**VertOffset** Lets you specify the vertical offset in tenths of a micron.
1	.**Opacity** Lets you specify the shadow opacity from 0 to 100.
2	.**Feather** Lets you specify the shadow feathering from 0 to 100.
3	.**FeatherType** 0=Inside
	4 1=Middle
	5 2=Outside
	6 3=Average
7	.**FeatherEdge** 0=Linear
	8 1=Square
	9 2=Flat
	10 3=Inverse square

Example

```
.StoreColor DRAW_COLORMODEL_RGB&, 0, 0, 0 'Make a black drop shadow
0   .ApplyDropShadow LENGTHCONVERT(LC_INCHES, LC_TENTHS_OFA_MICRON, 1), \
\LENGHTCONVERT(LC_INCHES, LC_TENTHS_OFA_MICRON, 1), 50, 100, 0, 0
```

The above example applies a drop shadow.

{button ,AL(`CLS_CorelScript;FNC_ApplyDropShadow')} **Related Topics**

CorelScript.Benchmark

Function **Benchmark**(ByVal **Enable** AS Boolean) AS Long

0 CorelScript

0 Member of **CorelScript**

Parameters	**Description**
Enable	Description of 'Enable' goes here

Return value

0 Returns Long

1 Example

2 Example of usage goes here

{button ,AL(`CLS_CorelScript;FNC_Benchmark')} **Related Topics**

CorelScript.GetPageCount

Function **GetPageCount**() AS Long

0 CorelScript

Description

This command returns the number of pages in the current document.

Parameter	Description
.**Name**	Returns the name of the current document.

{button ,AL(`CLS_CorelScript;FNC_GetPageCount')} **Related Topics**

CorelScript.CutToClipboard

Function **CutToClipboard**() AS Long

0 CorelScript

Description

This command removes the selected object(s) or text from your document and places a copy onto the Clipboard.

{button ,AL(`CLS_CorelScript;FNC_CutToClipboard')} **Related Topics**

CorelScript.PlaceInside

Function **PlaceInside**(ByVal **Index** AS Long, ByVal **XDisp** AS Long, ByVal **YDisp** AS Long, ByVal **ForceCenter** AS Boolean, ByVal **Center** AS Boolean) AS Long

0 CorelScript

Description

This command will PowerClip the current selection.

Parameter	Description
.**Index**	Lets you specify the object ID.
0	.**XDisp** Lets you specify the X coordinate in the object to PowerClip.
1	.**YDisp** Lets you specify the Y coordinate in the object to PowerClip.
2	.**ForceCenter** Set to TRUE (-1) will use the Center parameter which centers the object inside the container. Set to FALSE (0) Uses the workspace preferences.
3	.**Center** Set to TRUE (-1) center the objet inside the container.

{button ,AL(`CLS_CorelScript;FNC_PlaceInside')} **Related Topics**

CorelScript.CopyPowerClip

Function **CopyPowerClip**(ByVal **Index** AS Long, ByVal **XDisp** AS Long, ByVal **YDisp** AS Long) AS Long

0 CorelScript

Description

This command copies the PowerClip of the object to the current selection.

Parameter	Description
.**Index**	Lets you specify the object ID.
0	.**XDisp** Lets you specify the X coordinate of the PowerClip in the object to copy.
1	.**YDisp** Lets you specify the Y coordinate of the PowerClip in the object to copy.

{button ,AL(`CLS_CorelScript;FNC_CopyPowerClip')} **Related Topics**

CorelScript.ExtractContents

Function **ExtractContents**() AS Long

0 CorelScript

Description

This command extracts the contents of the selected PowerClip.

{button ,AL(`CLS_CorelScript;FNC_ExtractContents')} **Related Topics**

CorelScript.StartEditContents

Function **StartEditContents**() AS Long

0 CorelScript

Description

This command start to edit the contents of the selected PowerClip.

{button ,AL(`CLS_CorelScript;FNC_StartEditContents')} **Related Topics**

CorelScript.StopEditContents

Function **StopEditContents**() AS Long

0 CorelScript

Description

This command stops to edit the contents of the selected PowerClip.

{button ,AL(`CLS_CorelScript;FNC_StopEditContents')} **Related Topics**

CorelScript.UngroupAll

Function **UngroupAll**() AS Long

0 CorelScript

Description

This command breaks up all the groups into its individual objects.

Example

```
.UngroupAll
```

The above example breaks up all the grouped objects into its individual object components.

{button ,AL(`CLS_CorelScript;FNC_UngroupAll')} **Related Topics**

CorelScript.CreateGridBoxes

Function **CreateGridBoxes**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **Wide** AS Long, ByVal **High** AS Long) AS Long

0 CorelScript

Description

This command draws a grid box.

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
2	.**Right** Lets you specify the X-coordinate of the lower-right corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
3	.**Wide** Lets you specify the number of cells wide.
4	.**High** Lets you specify the number of cells high.

{button ,AL(`CLS_CorelScript;FNC_CreateGridBoxes')} **Related Topics**

CorelScript.CreateSpiral

Function **CreateSpiral**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long, ByVal **NumRevolutions** AS Long, ByVal **SpiralType** AS Long, ByVal **GrowthRate** AS Long) AS Long

0 CorelScript

Description

This command draws a spiral.

Parameter	Description
.**Top**	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
0	.**Left** Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
1	.**Bottom** Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
2	.**NumRevolutions** Lets you specify the number of revolutions in the spiral.
3	.**SpiralType** Lets you specify the type of spiral. 0 = Symmetrical 1 = Logarithmic
4	.**GrowthRate** Lets you specify the growth rate of the spiral.

{button ,AL(`CLS_CorelScript;FNC_CreateSpiral')} **Related Topics**

CorelScript.GetObjectData

Function **GetObjectData**(ByVal **FieldName** AS String) AS String

0 CorelScript

Description

This function returns a specified object data field of a selected object. If more than one object is selected, the function returns the specified object-data field of the last selected object.

Parameter	Description
ReturnString$	Returns the object data of the selected object.
0	.**FieldName** Lets you specify the name of object data field.

{button ,AL(`CLS_CorelScript;FNC_GetObjectData')} **Related Topics**

CorelScript.SetObjectData

Function **SetObjectData**(ByVal **FieldName** AS String, ByVal **FieldValue** AS String) AS Long

0 CorelScript

Description

This command lets you set object data values for selected objects.

Parameter	Description
.**FieldName**	Lets you specify the name of the object data field to set.
0 .**FieldValue**	Lets you specify the value of the object data field to set.

{button ,AL(`CLS_CorelScript;FNC_SetObjectData')} **Related Topics**

CorelScript.AddObjectDataField

Function **AddObjectDataField**(ByVal **FieldName** AS String) AS Long

0 CorelScript

Description

This function adds an object data field. The new object data field must be followed by its definition. See DefineObjectDataField.

Parameter	Description
.**FieldName**	Lets you specify the name of object data field.

{button ,AL(`CLS_CorelScript;FNC_AddObjectDataField')} **Related Topics**

CorelScript.DefineObjectDataField

Function **DefineObjectDataField**(ByVal **FieldName** AS String, ByVal **Format** AS String, ByVal **DrawDefault** AS Boolean, ByVal **DocDefault** AS Boolean, ByVal **SummarizeGroup** AS Boolean) AS Long

0 CorelScript

Description

This function defines the properties of an object data field.

Parameter	Description
.**FieldName**	Lets you specify the name of object data field.
0	.**Format** Lets you specify the name of the field format.
1	.**DrawDefault** Set to TRUE (-1) object is saved in Coreldrw.ini.
2	.**DocDefault** Set to TRUE (-1) object is added to all objects.
3	.**SummarizeGroup** Set to TRUE (-1) to summarize groups.

{button ,AL(`CLS_CorelScript;FNC_DefineObjectDataField')} **Related Topics**

CorelScript.AddEnvelopeEffect

Function **AddEnvelopeEffect**(ByVal **PresetNumber** AS Long, ByVal **MappingMode** AS Long, ByVal **KeepLines** AS Boolean) AS Long

0 CorelScript

Description

This command adds an envelope to the selected object.

Parameter	Description
.**PresetNumber**	Lets you specify the envelope preset number. Choose from 1 to 39.
0	.**MappingMode** Lets you specify the mapping mode. 0 = Horinzontal 1 = Original 2 = Putty 3 = Vertical
1	.**KeepLines** Set to TRUE (-1) keeps the lines of the source object. Set to FALSE (0) excludes the lines of the source object.

{button ,AL(`CLS_CorelScript;FNC_AddEnvelopeEffect')} **Related Topics**

CorelScript.ApplyUniformBitmapLens

Function **ApplyUniformBitmapLens**(ByVal **Starting** AS Long, ByVal **Operation** AS Long, ByVal **Freeze** AS Boolean) AS Long

0 CorelScript

Description

This command applies a uniform bitmap lens to the selected object.

Parameter	Description	
.**Starting**	Lets you specify the starting transparency.	
0	.**Operation**	Lets you specify which operation to perform on the selected object.
1	.**Freeze**	Set to TRUE (-1) freezes the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyUniformBitmapLens')} **Related Topics**

CorelScript.ApplyFountainBitmapLens

Function **ApplyFountainBitmapLens**(ByVal **Type** AS Long, ByVal **CenterX** AS Long, ByVal **CenterY** AS Long, ByVal **Angle** AS Long, ByVal **Steps** AS Long, ByVal **Padding** AS Long, ByVal **Blend** AS Long, ByVal **Rate** AS Long, ByVal **Starting** AS Long, ByVal **Operation** AS Long, ByVal **Freeze** AS Boolean) AS Long

0 CorelScript

Description

This command applies a fountain bitmap lens to the selected object.

Parameter		Description
.**Type**		Lets you specify the fill type: 0 = Linear 1 = Radial 2 = Conical 3 = Square
0	.**CenterX**	Lets you specify the horizontal offset to the center of the fill. Choose between -100 to 100.
1	.**CenterY**	Lets you specify the vertical offset to the center of the fill. Choose between -100 to 100.
2	.**Angle**	Lets you specify the angle of the fill in tenths of a degree.
3	.**Steps**	Lets you specify the number of steps in the fill.
4	.**Padding**	Lets you specify the amount of padding to apply to the fill.
5	.**Blend**	Lets you specify the blending type: 0 = Direct 1 = Rainbow clockwise 2 = Rainbow counterclocwise 3 = Custom
6	.**Rate**	Lets you specify the mid-point to apply between the fill colors. Choose from 0 to 100.
7	.**Starting**	Lets you specify the starting transparency.
8	.**Operation**	Lets you specify which operation to perform on the selected object.
9	.**Freeze**	Set to TRUE (-1) freezes the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyFountainBitmapLens')} **Related Topics**

CorelScript.ApplyTwoColorBitmapLens

Function **ApplyTwoColorBitmapLens**(ByVal **FileName** AS String, ByVal **TileWidth** AS Long, ByVal **TileHeight** AS Long, ByVal **FirstTileOffsetX** AS Long, ByVal **FirstTileOffsetY** AS Long, ByVal **RowOffset** AS Boolean, ByVal **RowColumnOffset** AS Long, ByVal **SeamlessTiling** AS Boolean, ByVal **ScaleWithObject** AS Boolean, ByVal **RotationAngle** AS Long, ByVal **SkewAngle** AS Long, ByVal **Starting** AS Long, ByVal **Ending** AS Long, ByVal **Operation** AS Long, ByVal **Freeze** AS Boolean) AS Long

0 CorelScript

Description

This command applies a two-color bitmap lens to the selected object.

Parameter	Description
.**FileName**	Lets you specify the path and filenameof the two-color fill file to use.
0	.**TileWidth** Lets you specify the width of the tile in units specified by ScaleWithObject command.
1	.**TileHeight** Lets you specify the height of the tile in units specifies by the ScaleWithObject command.
2	.**FirstTileOffsetX** Lets you specify the amount to offset the first tile horizontally.
3	.**FirstTileOffsetY** Lets you specify the amount to offset the first tile vertically.
4	.**RowOffset** Set to TRUE (-1) enables the row offset. Set to FALSE (0) enables the column offset.
5	.**RowColumnOffset** Lets you specify the amount to offset the row or column. Choose between 0 and 100.
6	.**SeamlessTiling** Set to TRUE (-1) enables seamless tiling.
7	.**ScaleWithObject** Set to TRUE (-1) sets the height/width units to a tenth of a micron. Set to FALSE (0) set the measurement as a percentage.
8	.**RotationAngle** Lets you specify the amount that the tile is rotated in millionths of degrees.
9	.**SkewAngle** Lets you specify the amount that the tile is skewed in millionths of degrees.
10	.**Starting** Lets you specify the starting transparency.
11	.**Operation** Lets you specify which operation to perform on the selected object.
12	.Freeze Set to TRUE (-1) freezes the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyTwoColorBitmapLens')} **Related Topics**

CorelScript.ApplyTextureBitmapLens

Function **ApplyTextureBitmapLens**(ByVal **TextureLibrary** AS String, ByVal **TextureName** AS String, ByVal **TextureStyle** AS String, ByVal **TextureWidth** AS Long, ByVal **TextureHeight** AS Long, ByVal **TextureOffsetX** AS Long, ByVal **TextureOffsetY** AS Long, ByVal **RowOffset** AS Boolean, ByVal **RowColumnOffset** AS Long, ByVal **ScaleWithObject** AS Boolean, ByVal **RotationAngle** AS Long, ByVal **SkewAngle** AS Long, ByVal **Starting** AS Long, ByVal **Ending** AS Long, ByVal **Operation** AS Long, ByVal **Freeze** AS Boolean) AS Long

0 CorelScript

Description

This command applies a texture bitmap lens to the selected object.

Parameter	Description
.**TextureLibrary**	Lets you specify the name of the texture library.
0	.**TextureName** Lets you specify the name of the texture.
1	.**TextureStyle** Lets you specify the name of the texture style.
2	.**TextureWidth** Lets you specify the width of the tile in tenths of a micron.
3	.**TextureHeight** Lets you specify the height of the tile in tenths of a micron.
4	.**TextureOffsetX** Lets you specify the amount to offset the texture horizontally.
5	.**TextureOffsetY** Lets you specify the amount to offset the texure vertically.
6	.**RowOffset** Set to TRUE (-1) enables the row offset. Set to FALSE (0) enables the column offset.
7	.**RowColumnOffset** Lets you specify the amount to offset the row or column in tenths of a micron.
8	.**ScaleWithObject** Set to TRUE (-1) transforms the fill when the object is transformed.
9	.**RotationAngle** Lets you specify the amount that the tile is rotated in millionths of degrees.
10	.**SkewAngle** Lets you specify the amount that the tile is skewed in millionths of degrees.
11	.**Starting** Lets you specify the starting transparency.
12	.**Operation** Lets you specify which operation to perform on the selected object.
13	.**Freeze** Set to TRUE (-1) freezes the object.

{button ,AL(`CLS_CorelScript;FNC_ApplyTextureBitmapLens')} **Related Topics**

CorelScript.CreateObjectDataField

Function **CreateObjectDataField**(ByVal **FieldName** AS String, ByVal **Format** AS String, ByVal **Order** AS Long, ByVal **ColumnWidth** AS Long, ByVal **DrawDefault** AS Boolean, ByVal **DocDefault** AS Boolean, ByVal **SummarizeGroup** AS Boolean) AS Long

0 CorelScript

Description

This function creates a new object data field.

Parameter	Description
.**FieldName**	Lets you specify the name of object data field.
0	.**Format** Lets you specify the format for the field.
1	.**Order** Lets you specify the position in the field.
2	.**ColumnDefault** Lets you specify the width of the column.
3	.**DrawDefault** Set to TRUE (-1) object is saved in Coreldrw.ini.
4	.**DocDefault** Set to TRUE (-1) object is added to all objects.
5	.**SummarizeGroup** Set to TRUE (-1) to summarize groups.

{button ,AL(`CLS_CorelScript;FNC_CreateObjectDataField')} **Related Topics**

CorelScript.DeleteObjectDataField

Function **DeleteObjectDataField**(ByVal **FieldName** AS String) AS Long

0 CorelScript

Description

This function deletes the object data field.

Parameter	Description
.**FieldName**	Lets you specify the name of object data field.

{button ,AL(`CLS_CorelScript;FNC_DeleteObjectDataField')} **Related Topics**

CorelScript.RenameObjectDataField

Function **RenameObjectDataField**(ByVal **OldFieldName** AS String, ByVal **NewFieldName** AS String) AS Long

0 CorelScript

Description

This command lets you rename an object data field.

Parameter	Description
.**OldFieldName**	Lets you specify the old name of the object data field.
0 .**NewFieldName**	Lets you specify the new name of the object data field.

{button ,AL(`CLS_CorelScript;FNC_RenameObjectDataField')} **Related Topics**

CorelScript.OrderObjectDataFields

Function **OrderObjectDataFields**(ByVal **FieldToMove** AS Long, ByVal **NewPosition** AS Long) AS Long

0 CorelScript

Description

This function changes the order of the object data fields.

Parameter	Description
.**FieldToMove**	Lets you specify the name of the field to move.
0	.**NewPosition** Lets you specify the position to place the field.

{button ,AL(`CLS_CorelScript;FNC_OrderObjectDataFields')} **Related Topics**

CorelScript.CopyObjectDataFields

Function **CopyObjectDataFields**(ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This function copies all data fields from the target object.

Parameter	Description
.**Index**	Lets you specify the object ID.

{button ,AL(`CLS_CorelScript;FNC_CopyObjectDataFields')} **Related Topics**

CorelScript.ClearAllObjectData

Function **ClearAllObjectData**() AS Long

0 CorelScript

Description

This function clears all the fields and data from the target object.

{button ,AL(`CLS_CorelScript;FNC_ClearAllObjectData')} **Related Topics**

CorelScript.ClearObjectData

Function **ClearObjectData**(ByVal **FieldName** AS String) AS Long

0 CorelScript

Description

This function clears a field and data from the selected object.

Parameter	Description
.**FieldName**	Lets you specify the name of the objects data field.

{button ,AL(`CLS_CorelScript;FNC_ClearObjectData')} **Related Topics**

CorelScript.PasteObjectData

Function **PasteObjectData**(ByVal **Index** AS Long, ByVal **FieldName** AS String) AS Long

0 CorelScript

Description

This function pastes the contents from the Clipboard into a field.

Parameter	Description
.**Index**	Lets you specify the object ID.
0	.**FieldName** Lets you specify the name of the object data field.

{button ,AL(`CLS_CorelScript;FNC_PasteObjectData')} **Related Topics**

CorelScript.RenameStyle

Function **RenameStyle**(ByVal **OldName** AS String, ByVal **NewName** AS String) AS Long

0 CorelScript

Description

This command renames a style name.

Parameter	Description
.**OldName**	Lets you specify the old name of the style.
0	.**NewName** Lets you specify the new name of the style.

{button ,AL(`CLS_CorelScript;FNC_RenameStyle')} **Related Topics**

CorelScript.CreateNewStyle

Function **CreateNewStyle**(ByVal **StyleType** AS Long, ByVal **StyleName** AS String) AS Long

0 CorelScript

Description

This command lets you create a new style.

Parameter	Description
.**StyleType**	Lets you specify the type of style: 0 = Artistic text 1 = Paragraph text 2= Graphic
0	.**StyleName** Lets you specify the name of the new style.

{button ,AL(`CLS_CorelScript;FNC_CreateNewStyle')} **Related Topics**

CorelScript.SaveStyleProp

Function **SaveStyleProp**(ByVal **StyleName** AS String, ByVal **UseFill** AS Boolean, ByVal **UseOutline** AS Boolean, ByVal **UseFont** AS Long, ByVal **UseAlignment** AS Long, ByVal **UseSpacing** AS Long, ByVal **UseLines** AS Long, ByVal **UseIndentsAndMargins** AS Long, ByVal **UseTextEffects** AS Long) AS Long

0 CorelScript

Description

This command lets you save the style based on the current properties.

Parameter	Description
.**StyleName**	Lets you specify the name of the new Style.
0	.**UseFill** Set to TRUE (-1) saves the fill as part of the style.
1	.**UseOutline** Set to TRUE (-1) saves the outline as part of the style.
2	.**UseFont** Lets you specify the font: 1 = Typeface 2 = Typestyle 4 = Size
3	.**UseAlignment** Specifices the alignement: 1 = Justification 2 = Tabs 4 = Hyphenation
4	.**UseSpacing** Lets you specify the spacing: 1 = InterChar Spacing 2 = InterWord Spacing 4 = Interline Spacing 8 = Spacing Before Paragraphs
5	.**UseLines** Lets you specify the lines: 1 = Underline 2 = Overline 4 = Strikeout
6	.**UseIndentsAndMargins** Lets you specify the indents and margins: 1 = Bullet Indent 2 = First Line Indent 4 = Rest of Lines Indent 8 = Right Margin
7	.**UseTextEffects** Lets you specify the text effect 1 = Superscript/Subscript 2 = Capitalization Effect 4= Bullet

{button ,AL(`CLS_CorelScript;FNC_SaveStyleProp')} **Related Topics**

CorelScript.MoveLayerTo

Function **MoveLayerTo**(ByVal **DestLayer** AS String, ByVal **BeforeDest** AS Boolean) AS Long

0 CorelScript

Description

This command moves the current layer to another position.

Parameter	Description
.**DestLayer**	Lets you specify the name of the destination layer.
0	.**BeforeDest** Set to TRUE (-1) places the layer before the destination layer.

{button ,AL(`CLS_CorelScript;FNC_MoveLayerTo')} **Related Topics**

CorelScript.ChangeLayerColor

Function **ChangeLayerColor**(ByVal **LayerName** AS String, ByVal **PageNum** AS Long) AS Long

0 CorelScript

Description

This command lets you set the outline color for the override color..

Parameter	Description
.**LayerName**	Lets you specify the name of the Layer.
0	.**PageNum** Lets you specify the pafe number: 0 = Master

{button ,AL(`CLS_CorelScript;FNC_ChangeLayerColor')} **Related Topics**

CorelScript.ConvertOutlineToObject

Function **ConvertOutlineToObject**() AS Long

0 CorelScript

Description

This command converts an object's outline into an object.

{button ,AL(`CLS_CorelScript;FNC_ConvertOutlineToObject')} **Related Topics**

CorelScript.GetCurrentPageOrientation

Function **GetCurrentPageOrientation**() AS Long

0 CorelScript

Description

This command returns the orientation of the current document page.

Parameter	Description
.**Orient**	Lets you specify the page orientation: 0 = Portrait 1 = Landscape

---

{button ,AL(`CLS_CorelScript;FNC_GetCurrentPageOrientation')} **Related Topics**

CorelScript.SetCurrentPageOrientation

Function **SetCurrentPageOrientation**(ByVal **Orient** AS Long) AS Long

0 CorelScript

Description

This command sets the orientation for the current page.

Parameter	Description
.**Orient**	Lets you specify the page orientation: 0 = Portrait 1 = Landscape

{button ,AL(`CLS_CorelScript;FNC_SetCurrentPageOrientation')} **Related Topics**

CorelScript.GetCurrentPageSize

Function **GetCurrentPageSize**(ByRef **Width** AS Long, ByRef **Height** AS Long) AS Long

0 CorelScript

Description

This command returns the width and height of the current document page.

Parameter	Description
.**Width**	Returns the width of the current page in tenths of a micron.
0 .**Height**	Returns the height of the current page in tenths of a micron.

{button ,AL(`CLS_CorelScript;FNC_GetCurrentPageSize')} **Related Topics**

CorelScript.SetCurrentPageSize

Function **SetCurrentPageSize**(ByVal **Width** AS Long, ByVal **Height** AS Long) AS Long

0 CorelScript

Description

This command sets the size for the current page.

Parameter	Description
.**Width**	Lets you specify the page width in tenths of a micron.
0 .**Height**	Lets you specify the page height in tenths of a micron.

{button ,AL(`CLS_CorelScript;FNC_SetCurrentPageSize')} **Related Topics**

CorelScript.EditLayer

Function **EditLayer**(ByVal **LayerName** AS String, ByVal **PageNum** AS Long, ByVal **NewName** AS String, ByVal **Visible** AS Boolean, ByVal **Printable** AS Boolean, ByVal **Locked** AS Boolean, ByVal **Master** AS Boolean, ByVal **WireOverride** AS Boolean, ByVal **ChangeColor** AS Boolean) AS Long

0 CorelScript

Description

This command sets all the layer properties.

Parameter	Description
.**LayerName**	Lets you specify the layer name.
0	.**PageNum** Lets you specify the page number.
1	,**NewName** Lets you specify the new name for the layer.
2	.**Visible** Set to TRUE (-1) makes the layer visible.
3	.**Printable** Set to TRUE (-1) makes the layer printable.
4	.**Locked** Set to TRUE (-1) makes the layer locked.
5	.**Master** Set to TRUE (-1) makes the layer the master layer.
6	.**WireOverride** Set to TRUE (-1) forces the view to wireframe display.
7	.**ChangeColor** Set to TRUE (-1) changes the color of the wireframe view.

{button ,AL(`CLS_CorelScript;FNC_EditLayer')} **Related Topics**

CorelScript.GetPolygonSides

Function **GetPolygonSides**() AS Long

0 CorelScript

Description

This function returns the number of sides in a polygon.

{button ,AL(`CLS_CorelScript;FNC_GetPolygonSides')} **Related Topics**

CorelScript.GetEllipseType

Function **GetEllipseType**() AS Long

0 CorelScript

Description

This function returns the type of ellipse. (0 = Ellipse, 1 = Arc, 2 = Pie)

{button ,AL(`CLS_CorelScript;FNC_GetEllipseType')} **Related Topics**

CorelScript.GetEllipseStartAngle

Function **GetEllipseStartAngle**() AS Double

0 CorelScript

Description

This function returns the start angle of the ellipse.

{button ,AL(`CLS_CorelScript;FNC_GetEllipseStartAngle')} **Related Topics**

CorelScript.GetEllipseEndAngle

Function **GetEllipseEndAngle**() AS Double

0 CorelScript

Description

This function returns the end angle of the ellipse.

{button ,AL(`CLS_CorelScript;FNC_GetEllipseEndAngle')} **Related Topics**

CorelScript.GetEllipseClockwise

Function **GetEllipseClockwise**() AS Boolean

0 CorelScript

Description

This function returns false of the ellipse is counterclockwise.

{button ,AL(`CLS_CorelScript;FNC_GetEllipseClockwise')} **Related Topics**

CorelScript.GetPolygonType

Function **GetPolygonType**() AS Long

0 CorelScript

Description

This function returns the type of polygon. (0 = Polyong, 1 = Star, 2 = Polygon as star)

{button ,AL(`CLS_CorelScript;FNC_GetPolygonType')} **Related Topics**

CorelScript.GetPolygonSharpness

Function **GetPolygonSharpness**() AS Long

0 CorelScript

Description

This function returns the sharpness value in a polygon.

{button ,AL(`CLS_CorelScript;FNC_GetPolygonSharpness')} **Related Topics**

CorelScript.GetObjectCount

Function **GetObjectCount**(ByVal **Selection** AS Boolean, ByVal **Grouped** AS Boolean) AS Long

0 CorelScript

Description

This function counts the number of objects.

Parameter	Description
.**Selection**	Set to TRUE (-1) counts objects in the current selection.
0 .**Grouped**	Set to TRUE (-1) counts groups as one object.

{button ,AL(`CLS_CorelScript;FNC_GetObjectCount')} **Related Topics**

CorelScript.GetObjectID

Function **GetObjectID**(ByVal **Index** AS Long, ByVal **Selection** AS Boolean, ByVal **Grouped** AS Boolean) AS Long

0 CorelScript

Description

This function returns the ID of the Index-th object.

Parameter		Description
.**Index**		Lets you specify the object ID. 0▸GetObjectCount-1
0	.**Selection**	Set to TRUE (-1) counts objects in the current selection.
1	.**Grouped**	Set to TRUE (-1) counts groups as one object.

{button ,AL(`CLS_CorelScript;FNC_GetObjectID')} **Related Topics**

CorelScript.IsSelection

Function **IsSelection**() AS Boolean

0 CorelScript

Description

This command returns true if an object is selected.

{button ,AL(`CLS_CorelScript;FNC_IsSelection')} **Related Topics**

CorelScript.IsDocument

Function **IsDocument**() AS Boolean

0 CorelScript

Description

This command returns true if a document is open.

{button ,AL(`CLS_CorelScript;FNC_IsDocument')} **Related Topics**

CorelScript.GetTextWordCount

Function **GetTextWordCount**() AS Long

0 CorelScript

Description

This command returns the number of words in your document.

{button ,AL(`CLS_CorelScript;FNC_GetTextWordCount')} **Related Topics**

CorelScript.GetTextFontSize

Function **GetTextFontSize**() AS Long

0 CorelScript

Description

This command returns the font size of the selected text.

{button ,AL(`CLS_CorelScript;FNC_GetTextFontSize')} **Related Topics**

CorelScript.GetTextFontName

Function **GetTextFontName**() AS String

0 CorelScript

Description

This command returns the font name of the selected text.

{button ,AL(`CLS_CorelScript;FNC_GetTextFontName')} **Related Topics**

CorelScript.GetTextString

Function **GetTextString**() AS String

0 CorelScript

Description

This command returns the actual text string.

{button ,AL(`CLS_CorelScript;FNC_GetTextString')} **Related Topics**

CorelScript.GetNodeSelectedCount

Function **GetNodeSelectedCount**() AS Long

0 CorelScript

Description

This command returns the number of nodes in the current selection.

{button ,AL(`CLS_CorelScript;FNC_GetNodeSelectedCount')} **Related Topics**

CorelScript.GetCurveFirstNodePosition

Function **GetCurveFirstNodePosition**(ByRef **X** AS Long, ByRef **Y** AS Long) AS Long

0 CorelScript

Description

This command returns the coordinates of the first node in the selected curve.

Parameter	Description
.**X**	Lets you specify the X coordinate.
0	.**Y** Lets you specify the Y coordinate.

{button ,AL(`CLS_CorelScript;FNC_GetCurveFirstNodePosition')} **Related Topics**

CorelScript.GetCurveLastNodePosition

Function **GetCurveLastNodePosition**(ByRef **X** AS Long, ByRef **Y** AS Long) AS Long

0 CorelScript

Description

This command returns the coordinates of the last node in the selected curve.

Parameter	Description
.**X**	Lets you specify the X coordinate.
0	.**Y** Lets you specify the Y coordinate.

{button ,AL(`CLS_CorelScript;FNC_GetCurveLastNodePosition')} **Related Topics**

CorelScript.GetCurveIthNodePosition

Function **GetCurveIthNodePosition**(ByVal **Index** AS Long, ByRef **X** AS Long, ByRef **Y** AS Long) AS Long

0 CorelScript

Description

This command returns the coordinates of the I-th node in the selected curve.

Parameter	Description
.**X**	Lets you specify the I-th coordinate.
0	.**Y** Lets you specify the I-th coordinate.

{button ,AL(`CLS_CorelScript;FNC_GetCurveIthNodePosition')} **Related Topics**

CorelScript.BeginCommandGroup

Function **BeginCommandGroup**(ByVal **UndoString** AS String) AS Long

0 CorelScript

Description

This command starts a group of commands to have a clean undo stack.

Parameter	Description
.**UndoString**	Lets you specify the string to be used.

{button ,AL(`CLS_CorelScript;FNC_BeginCommandGroup')} **Related Topics**

CorelScript.EndCommandGroup

Function **EndCommandGroup**() AS Long

0 CorelScript

Description

This command ends a group of commands.

{button ,AL(`CLS_CorelScript;FNC_EndCommandGroup')} **Related Topics**

CorelScript.GetCurveClose

Function **GetCurveClose**() AS Long

0 CorelScript

Description

This command returns true if the curve is closed.

{button ,AL(`CLS_CorelScript;FNC_GetCurveClose')} **Related Topics**

CorelScript.GetRectangleRadius

Function **GetRectangleRadius**(ByRef **Radius1** AS Double, ByRef **Radius2** AS Double, ByRef **Radius3** AS Double, ByRef **Radius4** AS Double) AS Long

0 CorelScript

Description

This function returns the radius for each of a rectangle's corners.

Parameter	Description
.**Radius1**	Lets you specify the upper left radius of the rectangle.
0 .**Radius2**	Lets you specify the upper right radius of the rectangle.
1 .**Radius3**	Lets you specify the lower right radius of the rectangle.
2 .**Radius4**	Lets you specify the lower right radius of the rectangle.

{button ,AL(`CLS_CorelScript;FNC_GetRectangleRadius')} **Related Topics**

CorelScript.ConvertToBitmap

Function **ConvertToBitmap**(ByVal **BitDepth** AS Long, ByVal **Grayscale** AS Boolean, ByVal **Dithered** AS Boolean, ByVal **TransparentBG** AS Boolean, ByVal **Resolution** AS Long, ByVal **AntiAliasing** AS Long, ByVal **UseColorProfile** AS Boolean) AS Long

0 CorelScript

Description

This command converts a vector object to a bitmap.

Parameter	Description
.**BitDepth**	Lets you specify bit depth.
0	.**Grayscale** Set to TRUE (-1) convert to grayscale.
1	.**Dithered** Set to TRUE (-1) enables dithering.
2	.**TransparentBG** Set to TRUE (-1) enables transparent background.
3	.**Resolution** Lets you specify the resolution.
4	.**AntiAliasing** Lets you specify the anti aliasing: 0 = None 1 = Normal 2 = Super Sampling
5	.**UseColorProfile** Set to TRUE (-1) use the color profile.

{button ,AL(`CLS_CorelScript;FNC_ConvertToBitmap')} **Related Topics**

CorelScript.InflateBitmap

Function **InflateBitmap**(ByVal **Width** AS Long, ByVal **Height** AS Long, ByVal **InflationType** AS Long) AS Long

0 CorelScript

Description

This command inflates a bitmap.

Parameter	Description
.**Width**	Lets you specify the bitmap width.
0	.**Height** Lets you specify the bitmap height.
1	.**InflationType** Lets you specify the inflation type: 0 = Size 1 = incrementation 2 = Percentage

---

{button ,AL(`CLS_CorelScript;FNC_InflateBitmap')} **Related Topics**

CorelScript.GetBitmapResolution

Function **GetBitmapResolution**(ByRef **XRes** AS Long, ByRef **YRes** AS Long) AS Long

0 CorelScript

Description

This command returns the bitmap resolution.

Parameter	Description
.**XRes**	Lets you specify the horizontal resolution.
0 .**YRes**	Lets you specify the vertical resolution.

{button ,AL(`CLS_CorelScript;FNC_GetBitmapResolution')} **Related Topics**

CorelScript.GetBitmapSize

Function **GetBitmapSize**(ByRef **Width** AS Long, ByRef **Height** AS Long) AS Long

0 CorelScript

Description

This command returns the bitmap size.

Parameter	Description
.**Width**	Lets you specify the bitmap width.
0 .**Height**	Lets you specify the bitmap height.

{button ,AL(`CLS_CorelScript;FNC_GetBitmapSize')} **Related Topics**

CorelScript.ConvertBitmapTo

Function **ConvertBitmapTo**(ByVal **EffectID** AS Long) AS Long

0 CorelScript

Description

This command converts the selected bitmap to another bit depth.

Parameter	Description
.**EffectID**	Lets you specify ID.

{button ,AL(`CLS_CorelScript;FNC_ConvertBitmapTo')} **Related Topics**

CorelScript.ApplyBitmapEffect

Function **ApplyBitmapEffect**(ByVal **EffectID** AS Long) AS Long

0 CorelScript

Description

This command applies the bitmap effect to the selected image.

Parameter	Description
.**EffectID**	Lets you specify effect ID.

{button ,AL(`CLS_CorelScript;FNC_ApplyBitmapEffect')} **Related Topics**

CorelScript.IsBitmapExternallyLinked

Function **IsBitmapExternallyLinked**() AS Boolean

0 CorelScript

0 Member of **CorelScript**

1 Return value

2 Returns Boolean

3 Example

4 Example of usage goes here

{button ,AL(`CLS_CorelScript;FNC_IsBitmapExternallyLinked')} **Related Topics**

CorelScript.ResolveBitmapLink

Function **ResolveBitmapLink**() AS Long

0 CorelScript

Description

This command resolves any external links.

{button ,AL(`CLS_CorelScript;FNC_ResolveBitmapLink')} **Related Topics**

CorelScript.UpdateBitmapLink

Function **UpdateBitmapLink**() AS Long

0 CorelScript

Description

This command updates any external links.

{button ,AL(`CLS_CorelScript;FNC_UpdateBitmapLink')} **Related Topics**

CorelScript.ResolveAllBitmapsLink

Function **ResolveAllBitmapsLink**() AS Long

0 CorelScript

Description

This command resolves all external links.

{button ,AL(`CLS_CorelScript;FNC_ResolveAllBitmapsLink')} **Related Topics**

CorelScript.GetColor

Function **GetColor**(ByVal **StoreColor** AS Boolean, ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef **V2** AS Long, ByRef **V3** AS Long, ByRef **V4** AS Long, ByRef **V5** AS Long, ByRef **V6** AS Long, ByRef **V7** AS Long) AS Long

0 CorelScript

Description

This function gets a color from the color dialog box.

Parameter	Description
.**StoreColor**	-1 stores the color 0 does not store the color
0 .**ColorModel** (optional)	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value.
1	Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
2	.**Color1** (optional) Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
3	.**Color2** (optional) Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
4	.**Color3** (optional) Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
5	.**Color4** (optional) Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
6	.**V5** (optional) Lets you specify the fifth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
7	.**V6** (optional) Lets you specify the sixth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
8	.**V7** (optional) Lets you specify the seventh color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

{button ,AL(`CLS_CorelScript;FNC_GetColor')} **Related Topics**

CorelScript.GetCurveNodeCount

Function **GetCurveNodeCount**() AS Long

0 CorelScript

Description

This command returns the number of nodes of the selected curve.

{button ,AL(`CLS_CorelScript;FNC_GetCurveNodeCount')} **Related Topics**

CorelScript.GetCurveSubpathCount

Function **GetCurveSubpathCount**() AS Long

0 CorelScript

Description

This command returns the number of subpaths of the selected curve.

{button ,AL(`CLS_CorelScript;FNC_GetCurveSubpathCount')} **Related Topics**

CorelScript.SetErrorHandling

Function **SetErrorHandling**(ByVal **Msg** AS Boolean) AS Boolean

0 CorelScript

Description

This command sets how the errors should be returned.

Parameter	Description
.**Msg**	Set to TRUE (-1) returns error messages. Set to FALSE (0) returns error code.

{button ,AL(`CLS_CorelScript;FNC_SetErrorHandling')} **Related Topics**

CorelScript.GetCurveLength

Function **GetCurveLength**() AS Long

0 CorelScript

Description

This command returns the curve length.

{button ,AL(`CLS_CorelScript;FNC_GetCurveLength')} **Related Topics**

CorelScript.SelectObjectOfType

Function **SelectObjectOfType**(ByVal **Type** AS Long) AS Long

0 CorelScript

Description

This command selects the object of a given type.

Parameter	Description
.**Type**	Lets you specify the type of object to select.

{button ,AL(`CLS_CorelScript;FNC_SelectObjectOfType')} **Related Topics**

CorelScript.GetCurrentPageName

Function **GetCurrentPageName**() AS String

0 CorelScript

Description

This command returns the name of the current page.

{button ,AL(`CLS_CorelScript;FNC_GetCurrentPageName')} **Related Topics**

CorelScript.SetCurrentPageName

Function **SetCurrentPageName**(ByVal **Name** AS String) AS Long

0 CorelScript

Description

This command sets the current page name.

Parameter	Description
.**Name**	Lets you specify the name for the current page.

{button ,AL(`CLS_CorelScript;FNC_SetCurrentPageName')} **Related Topics**

CorelScript.GetUserClick

Function **GetUserClick**(ByRef **XCoord** AS Long, ByRef **YCoord** AS Long, ByVal **TimeOut** AS Long, ByVal **IgnoreSnap** AS Boolean) AS Long

0 CorelScript

Description

This function returns the coordinates of a user click.

Parameter	Description
.**XCoord**	Returns the X coordinate of the selection click.
0	.**YCoord** Returns the Y coordinate of the selection click.
1	.**Timeout** (optional) Lets you specify the amount of time, in seconds, to wait for the user to click. Defaults to 10.
2	.**IgnoreSnap** (optional) Defaults to TRUE (-1) 0 does not ignore the snap -1

ignores the snap

{button ,AL(`CLS_CorelScript;FNC_GetUserClick')} **Related Topics**

CorelScript.ZoomToSelection

Function **ZoomToSelection**() AS Long

0 CorelScript

Description

This command changes the view by zooming in on the current selection.

{button ,AL(`CLS_CorelScript;FNC_ZoomToSelection')} **Related Topics**

CorelScript.ZoomToAllObjects

Function **ZoomToAllObjects**() AS Long

0 CorelScript

Description

This command changes the view by zooming to all the objects on the page.

{button ,AL(`CLS_CorelScript;FNC_ZoomToAllObjects')} **Related Topics**

CorelScript.ZoomToPage

Function **ZoomToPage**() AS Long

0 CorelScript

Description

This command changes the view by zooming to the page.

{button ,AL(`CLS_CorelScript;FNC_ZoomToPage')} **Related Topics**

CorelScript.ZoomToWidth

Function **ZoomToWidth**() AS Long

0 CorelScript

Description

This command changes the view by zooming in on the page width.

{button ,AL(`CLS_CorelScript;FNC_ZoomToWidth')} **Related Topics**

CorelScript.ZoomToHeight

Function **ZoomToHeight**() AS Long

0 CorelScript

Description

This command changes the view by zooming in to the page height.

{button ,AL(`CLS_CorelScript;FNC_ZoomToHeight')} **Related Topics**

CorelScript.ZoomToRectangle

Function **ZoomToRectangle**(ByVal **Top** AS Long, ByVal **Left** AS Long, ByVal **Bottom** AS Long, ByVal **Right** AS Long) AS Long

0 CorelScript

Description

This command changes the view by zooming in on a specified area.

Parameter		Description
.**Top**		Lets you specify the Y coordinate of the top of the rectangle in tenths of a micron.
0	.**Left**	Lets you specify the X coordinate of the left of the rectangle in tenths of a micron.
1	.**Bottom**	Lets you specify the Y coordinate of the bottom of the rectangle in tenths of a micron.
2	.**Right**	Lets you specify the X coordinate of the right of the rectangle in tenths of a micron.

{button ,AL(`CLS_CorelScript;FNC_ZoomToRectangle')} **Related Topics**

CorelScript.ZoomIn

Function **ZoomIn**() AS Long

0 CorelScript

Description

This command changes the view by zooming in twice the size.

{button ,AL(`CLS_CorelScript;FNC_ZoomIn')} **Related Topics**

CorelScript.ZoomOut

Function **ZoomOut**() AS Long

0 CorelScript

Description

This command changes the view by zooming out half the size.

{button ,AL(`CLS_CorelScript;FNC_ZoomOut')} **Related Topics**

CorelScript.GetCDRFileVersion

Function **GetCDRFileVersion**(ByVal **FileName** AS String) AS Long

0 CorelScript

Description

This command returns the file version of the current document.

Parameter	Description
.**Filename**	Returns the file name for the current document.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileVersion')} **Related Topics**

CorelScript.GetCDRFileNotes

Function **GetCDRFileNotes**(ByVal **FileName** AS String) AS String

0 CorelScript

Description

This command returns the notes of the current document.

Parameter	Description
.**Filename**	Returns the notes for the current document.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileNotes')} **Related Topics**

CorelScript.GetCDRFileKeywords

Function **GetCDRFileKeywords**(ByVal **FileName** AS String) AS String

0 CorelScript

Description

This command returns the keywords of the current document.

Parameter	Description
.**Filename**	Returns the keywords for the current document.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileKeywords')} **Related Topics**

CorelScript.GetCDRFileCompRatio

Function **GetCDRFileCompRatio**(ByVal **FileName** AS String) AS Long

0 CorelScript

Description

This command returns the file compression ratio of the current document.

Parameter	Description
.**Filename**	Returns the file compression ratio for the current document.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileCompRatio')} **Related Topics**

CorelScript.GetCDRFileLastSavedBy

Function **GetCDRFileLastSavedBy**(ByVal **FileName** AS String) AS String

0 CorelScript

Description

This command returns the who saved the file last for the current document.

Parameter	Description
.**Filename**	Returns the last saved by name for the current document.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileLastSavedBy')} **Related Topics**

CorelScript.GetCDRFileThumbnail

Function **GetCDRFileThumbnail**(ByVal **CDRFileName** AS String, ByVal **BMPFileName** AS String) AS Long

0 CorelScript

Description

This command extracts the thumbnail of the CorelDRAW file and returns a BMP file.

Parameter	Description
.**CDRFilename**	Lets you specify the CorelDRAW file name from which to extract the thumbnail.
0 .**BMPFilename**	Returns the thumbnail image for the specified CorelDRAW file name.

{button ,AL(`CLS_CorelScript;FNC_GetCDRFileThumbnail')} **Related Topics**

CorelScript.SelectNode

Function **SelectNode**(ByVal **Index** AS Long, ByVal **AddToSelection** AS Boolean) AS Long

0 CorelScript

Description

This command lets you select the I-th node of the current curve.

Parameter	Description
.**Index**	Lets you specify the I-th node of the current curve.
0	.**AddToSelection** Set to TRUE (-1) adds the specified node to the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_SelectNode')} **Related Topics**

CorelScript.SelectNodeAt

Function **SelectNodeAt**(ByVal **X** AS Long, ByVal **Y** AS Long, ByVal **AddToSelection** AS Boolean) AS Long

0 CorelScript

Description

This command lets you select a specific node.

Parameter	Description
.**X**	Lets you specify the X coordinate of the node in the current curve.
0	**.Y** Lets you specify the Y coordinate of the node in the current curve.
1	.**AddToSelection** Set to TRUE (-1) adds the specified node to the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_SelectNodeAt')} **Related Topics**

CorelScript.SelectNextNode

Function **SelectNextNode**() AS Long

0 CorelScript

Description

This command selects the next node in the current curve.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_SelectNextNode')} **Related Topics**

CorelScript.DeleteNode

Function **DeleteNode**() AS Long

0 CorelScript

Description

This command deletes the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_DeleteNode')} **Related Topics**

CorelScript.ClearNodeSelection

Function **ClearNodeSelection**() AS Long

0 CorelScript

Description

This command clears the selection of nodes.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_ClearNodeSelection')} **Related Topics**

CorelScript.MoveNode

Function **MoveNode**(ByVal **DeltaX** AS Long, ByVal **DeltaY** AS Long) AS Long

0 CorelScript

Description

This command moves a node a specifies distance.

Parameter	Description
.**DeltaX**	Lets you specify the amount to move the node horizontally.
0 .**DeltaY**	Lets you specify the amount to move the node vertically.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_MoveNode')} **Related Topics**

CorelScript.GetNodeIndex

Function **GetNodeIndex**(ByVal **Position** AS Long) AS Long

0 CorelScript

Description

This command returns the index of the current node.

Parameter	Description
.**Position**	Returns the position index of the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_GetNodeIndex')} **Related Topics**

CorelScript.GetNodePosition

Function **GetNodePosition**(ByRef **X** AS Long, ByRef **Y** AS Long, ByVal **Position** AS Long) AS Long

0 CorelScript

Description

This command returns the index of the current node.

Parameter	Description
.X	Lets you specify the X coordinate of the current node.
0	**.Y** Lets you specify the Y coordinate of the current node.
1	.**Position** Lets you specify the position of the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_GetNodePosition')} **Related Topics**

CorelScript.AddNode

Function **AddNode**(ByVal **X** AS Long, ByVal **Y** AS Long) AS Long

0 CorelScript

Description

This command adds a node to the current curve at a specified location.

Parameter	Description
.X	Lets you specify the X coordinate.
0	**.Y** Lets you specify the Y coordinate.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_AddNode')} **Related Topics**

CorelScript.GetNodeType

Function **GetNodeType**() AS Long

0 CorelScript

Description

This command returns the type of node in the current selection.

Return Value

Returns of the following values:

- 0 Cusp
- 1 Smooth
- 2 Symmetrical

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_GetNodeType')} **Related Topics**

CorelScript.SetNodeType

Function **SetNodeType**(ByVal **Type** AS Long) AS Long

0 CorelScript

Description

This command sets the type of node.

Parameter	Description
.**Type**	Lets you specify the type of node: 0 = Cusp 1 = Smooth 2 = Symmetrical

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_SetNodeType')} **Related Topics**

CorelScript.GetSegmentType

Function **GetSegmentType**() AS Long

0 CorelScript

Description

This command returns the segment type for the current selection.

Return Value

Returns one of the following values:

- 0 Line
- 1 Curve

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_GetSegmentType')} **Related Topics**

CorelScript.SetSegmentType

Function **SetSegmentType**(ByVal **Type** AS Long) AS Long

0 CorelScript

Description

This command sets the type of segment.

Parameter	Description
.**Type**	Lets you specify the type of node: 0 = Line 1 = Curve

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_SetSegmentType')} **Related Topics**

CorelScript.GetSegmentLength

Function **GetSegmentLength**() AS Long

0 CorelScript

Description

This command returns the segment length for the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`CLS_CorelScript;FNC_GetSegmentLength')} **Related Topics**

CorelScript.SetCurrentWorkspace

Function **SetCurrentWorkspace**(ByVal **Name** AS String) AS Long

0 CorelScript

Description

This command saves the current CorelDRAW workspace settings.

Parameter	Description
.**Name**	Lets you specify the workspace name to save for the current settings.

{button ,AL(`CLS_CorelScript;FNC_SetCurrentWorkspace')} **Related Topics**

CorelScript.GetCurrentWorkspaceName

Function **GetCurrentWorkspaceName**() AS String

0 CorelScript

Description

This command returns the current workspace name.

{button ,AL(`CLS_CorelScript;FNC_GetCurrentWorkspaceName')} **Related Topics**

CorelScript.GetWorkspaceDescription

Function **GetWorkspaceDescription**(ByVal **Name** AS String) AS String

0 CorelScript

Description

This command returns the workspace description.

Parameter	Description
.**Name**	Lets you specify the workspace name for which you want the description.

{button ,AL(`CLS_CorelScript;FNC_GetWorkspaceDescription')} **Related Topics**

CorelScript.IsDefaultWorkspace

Function **IsDefaultWorkspace**(ByVal **Name** AS String) AS Long

0 CorelScript

Description

This command specifies the default workspace.

Parameter	Description
.**Name**	Lets you specify the workspace name that you want to make the default workspace.

{button ,AL(`CLS_CorelScript;FNC_IsDefaultWorkspace')} **Related Topics**

CorelScript.GetWorkspaceCount

Function **GetWorkspaceCount**() AS Long

0 CorelScript

Description

This command returns the number of workspaces.

{button ,AL(`CLS_CorelScript;FNC_GetWorkspaceCount')} **Related Topics**

CorelScript.GetWorkspaceName

Function **GetWorkspaceName**(ByVal **Index** AS Long) AS String

0 CorelScript

Description

This command returns the name of the I-th workspace.

Parameter	Description
.**Index**	Lets you specify the workspace index number for which you want the name.

{button ,AL(`CLS_CorelScript;FNC_GetWorkspaceName')} **Related Topics**

CorelScript.CreatePaletteFromDocument

Function **CreatePaletteFromDocument**(ByVal **FileName** AS String, ByVal **Overwrite** AS Boolean) AS Long

0 CorelScript

Description

This command creates a color palette from the current document.

Parameter	Description
.**Filename**	Lets you specify the file name for the color palette.
0 .**Overwrite**	Set to TRUE (-1) overwrites the specifief file name.

{button ,AL(`CLS_CorelScript;FNC_CreatePaletteFromDocument')} **Related Topics**

CorelScript.CreatePaletteFromSelection

Function **CreatePaletteFromSelection**(ByVal **FileName** AS String, ByVal **Overwrite** AS Boolean) AS Long

0 CorelScript

Description

This command creates a color palette from the current selection.

Parameter	Description
.**Filename**	Lets you specify the file name for the color palette.
0 .**Overwrite**	Set to TRUE (-1) overwrites the specifief file name.

{button ,AL(`CLS_CorelScript;FNC_CreatePaletteFromSelection')} **Related Topics**

CorelScript.LoadPalette

Function **LoadPalette**(ByVal **FileName** AS String) AS Long

0 CorelScript

Description

This command loads the specifies color palette in to the current document.

Parameter	Description
.**Filename**	Lets you specify the file name of the color palette you want to load in to your document.

{button ,AL(`CLS_CorelScript;FNC_LoadPalette')} **Related Topics**

CorelScript.SavePalette

Function **SavePalette**(ByVal **FileName** AS String, ByVal **Overwrite** AS Boolean) AS Long

0 CorelScript

Description

This command saves the current color palette.

Parameter	Description
.**Filename**	Lets you specify the file name for the color palette.
0 .**Overwrite**	Set to TRUE (-1) overwrites the existing palette.

{button ,AL(`CLS_CorelScript;FNC_SavePalette')} **Related Topics**

CorelScript.GetCurrentPaletteName

Function **GetCurrentPaletteName**() AS String

0 CorelScript

Description

This function returns the name of the current color palette.

{button ,AL(`CLS_CorelScript;FNC_GetCurrentPaletteName')} **Related Topics**

CorelScript.GetPaletteColorCount

Function **GetPaletteColorCount**() AS Long

0 CorelScript

Description

This function returns the number of colors in the default Color Palette.

{button ,AL(`CLS_CorelScript;FNC_GetPaletteColorCount')} **Related Topics**

CorelScript.GetPaletteColor

Function **GetPaletteColor**(ByVal **Index** AS Long, ByRef **ColorModel** AS Long, ByRef **V1** AS Long, ByRef **V2** AS Long, ByRef **V3** AS Long, ByRef **V4** AS Long, ByRef **V5** AS Long, ByRef **V6** AS Long, ByRef **V7** AS Long) AS Long

0 CorelScript

Description

This function returns the palette color attributes of a selected object.

Parameter	Description
.**Index**	Lets you specify the color index.
0	.**ColorModel** Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome 1 To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 2 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
3	.**Color1** Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color valid value ranges.
4	.**Color2** Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
5	.**Color3** Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
6	.**Color4** Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
7	.**Color5** Lets you specify the fifth color component for .ColorModel.
8	.**Color6** Lets you specify the sixth color component for .ColorModel.
9	.**Color7** Lets you specify the seventh color component (density)

{button ,AL(`CLS_CorelScript;FNC_GetPaletteColor')} **Related Topics**

CorelScript.GetPaletteColorName

Function **GetPaletteColorName**(ByVal **Index** AS Long) AS String

0 CorelScript

Description

This command retrieves the Color name for a specific index.

Parameter	Description
.**Index**	Lets you specify the index of the color for which you want to retrieve the name.

{button ,AL(`CLS_CorelScript;FNC_GetPaletteColorName')} **Related Topics**

CorelScript.InsertPaletteColor

Function **InsertPaletteColor**(ByVal **Index** AS Long, ByVal **Name** AS String, ByVal **ColorModel** AS Long, ByVal **V1** AS Long, ByVal **V2** AS Long, ByVal **V3** AS Long, ByVal **V4** AS Long, ByVal **V5** AS Long, ByVal **V6** AS Long, ByVal **V7** AS Long) AS Long

0 CorelScript

Description

This command inserts a color into the default palette.

Parameter	Description
.**Index**	Lets you specify the color index.
0	.**Name** Lets you specify the color name.
1	.**ColorModel** Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome 2 To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 3 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
4	.**Color1** Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
5	.**Color2** Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
6	.**Color3** Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
7	.**Color4** Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
8	.**Color5** Lets you specify the fifth color component for .ColorModel.
9	.**Color6** Lets you specify the sixth color component for .ColorModel.
10	.**Color7** Lets you specify the seventh color component (density)

{button ,AL(`CLS_CorelScript;FNC_InsertPaletteColor')} **Related Topics**

CorelScript.DeletePaletteColor

Function **DeletePaletteColor**(ByVal **Index** AS Long) AS Long

0 CorelScript

Description

This command deletes the palette color from the default palette.

Parameter	Description
.**Index**	Lets you specify the color index that you want to delete.

{button ,AL(`CLS_CorelScript;FNC_DeletePaletteColor')} **Related Topics**

CorelScript.ReplacePaletteColor

Function **ReplacePaletteColor**(ByVal **Index** AS Long, ByVal **Name** AS String, ByVal **ColorModel** AS Long, ByVal **V1** AS Long, ByVal **V2** AS Long, ByVal **V3** AS Long, ByVal **V4** AS Long, ByVal **V5** AS Long, ByVal **V6** AS Long, ByVal **V7** AS Long) AS Long

0 CorelScript

Description

This command replaces the color in the default palette.

Parameter	Description
.**Index**	Lets you specify the color index.
0	.**Name** Lets you specify the color name.
1	.**ColorModel** Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome 2 To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. 3 Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
4	.**Color1** Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click color for valid value ranges.
5	.**Color2** Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
6	.**Color3** Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
7	.**Color4** Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click color for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
8	.**Color5** Lets you specify the fifth color component for .ColorModel.
9	.**Color6** Lets you specify the sixth color component for .ColorModel.
10	.**Color7** Lets you specify the seventh color component (density)

{button ,AL(`CLS_CorelScript;FNC_ReplacePaletteColor')} **Related Topics**

CorelScriptTools methods

CorelScriptTools Legend

- AngleConvert
- ASin
- BeginWaitCursor
- BuildDate
- BuildTime
- Dec
- EndWaitCursor
- FileAttr
- FindFirstFolder
- FindNextFolder
- FormatTime
- FromCentimeters
- FromCiceros
- FromDidots
- FromInches
- FromPicas
- FromPoints
- GetAppHandle
- GetColor
- GetCommandLine
- GetCurrFolder
- GetDateInfo
- GetFileBox
- GetFolder
- GetFont
- GetProcessInfo
- GetScriptFolder
- GetTempFolder
- GetTimeInfo
- GetType
- GetVersion
- GetWinHandle
- Kill
- LengthConvert
- Log
- MkFolder
- RegistryQuery
- Rename
- RmFolder
- ToCentimeters
- ToCiceros
- ToDidots
- ToInches
- ToPicas
- ToPoints

CorelScriptTools

Class **CorelScriptTools**

Methods Referenced by

The **CorelScriptTools** class defines the characteristics of Corel Script objects and describes the look and behavior of the objects through its properties and methods.

0 Corel SCRIPT is an application included with some Corel applications to record, play, and run recordings and scripts. Click for a list of Corel applications that support Corel SCRIPT.

1 Corel SCRIPT starts automatically when you record or play a recording from a Corel application that supports scripts. If you run a script for an application that is not running, Corel SCRIPT automatically starts the application.

2 A typical user can take advantage of Corel SCRIPT to automate repetitive tasks with scripts written in the Corel SCRIPT programming language. However, Corel SCRIPT is more than just a language that you can use to create and run application script files; it's also a powerful programming language that can be used as a standalone application.

3 The Corel SCRIPT programming language is based on the BASIC programming language. If you're already familiar with a version of BASIC, you'll find the Corel SCRIPT programming language easy to read and understand.

4 A script is a computer program that executes a series of instructions with a single command. Generally, scripts are used to automate repetitive tasks, or simplify complicated actions; they can also be used to prompt for user input, display messages, and interact with other applications.

5 Scripts can significantly increase your productivity with Corel applications such as CorelDRAW or VENTURA, by automating repetitive tasks. For example, a script could be used to open a group of files, perform a set of editing actions, or set an application's default properties. In their simplest form, scripts replicate a Corel application's keystrokes, and toolbar, menu, and mouse commands. In a more complex form, scripts can include the commands and constructs of a programming language. For example, you could create a script that only replicates an application's commands once a series of logical requirements have been met.

{button ,AL(`CLS_CorelScriptTools')} Related Topics

CorelScriptTools.AngleConvert

Function **AngleConvert**(ByVal **FromUnit** AS Long, ByVal **ToUnit** AS Long, ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a number from one angle measurement to another.

Parameters	Description
FromUnit	Any number from 1 to 5 that indicates the unit of measurement from which to convert. 1 = degrees 2 = radians 3 = gradients 4 = Corel PHOTO-PAINT degrees (tenths of a degree) 5 = CorelDRAW degrees (millionths of a degree)
ToUnit	Any number from 1 to 5 that indicates the unit of measurement to convert to. 1 = degrees 2 = radians 3 = gradients 4 = Corel PHOTO-PAINT degrees (tenths of a degree) 5 = CorelDRAW degrees (millionths of a degree)
Value	Any numeric expressions specifying the value to be converted.

Example

```
x_rads = ANGLECONVERT(1, 2, 90)
```

The above example converts 90 degrees to radians. The variable **x_rads** equals 1.57142857142932.

{button ,AL(`CLS_CorelScriptTools;FNC_AngleConvert')} **Related Topics**

CorelScriptTools.ASin

Function **ASin**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Returns the inverse sine (arc sine) of a given value. The result is an angle in radians bounded by -π /2 and π /2.

Parameter	Description
x	A numeric expression between -1 and 1.

Note

- To convert the result from radians to degrees, use the ANGLECONVERT function or multiply the result by 180/3.14152.

Examples

```
w = ASIN(-0.75)
0   x = ASIN(0.75)
1   y = ASIN(0)
2   z = ASIN(1)
```

In the above example, **w** is equal to -0.8480621, **x** is equal to 0.8480621, **y** is equal to 0, and **z** is equal to 1.570796.

{button ,AL(`CLS_CorelScriptTools;FNC_ASin')} **Related Topics**

CorelScriptTools.BeginWaitCursor

Sub **BeginWaitCursor**()

CorelScriptTools

Description

The BEGINWAITCURSOR statement sets the mouse pointer to Busy. The Busy pointer usually appears as an hourglass on most systems. This command is useful for scripts that perform long operations. By setting the pointer to Busy, the user is alerted that the script is still executing.

0 The ENDWAITCURSOR statement returns the pointer to its normal state. If the ENDWAITCURSOR statement is not in a script that uses the BEGINWAITCURSOR statement, the pointer reverts to its normal state after the script finishes executing. It is good programming practice to use an ENDWAITCURSOR statement with every BEGINWAITCURSOR statement.

Example

```
BEGINWAITCURSOR
WAIT FOR 30
0   ENDWAITCURSOR
```

In the above example, the pointer is set to Busy, script execution is paused for 30 seconds, and then the pointer reverts to its normal state.

{button ,AL(`CLS_CorelScriptTools;FNC_BeginWaitCursor')} **Related Topics**

CorelScriptTools.EndWaitCursor

Sub **EndWaitCursor**()

CorelScriptTools

Description

The BEGINWAITCURSOR statement sets the mouse pointer to Busy. The Busy pointer usually appears as an hour-glass on most systems. This command is useful for scripts that perform long operations. By setting the pointer to Busy, the user is alerted that the script is still executing.

0 The ENDWAITCURSOR statement returns the pointer to its normal state. If the ENDWAITCURSOR statement is not in a script that uses the BEGINWAITCURSOR statement, the pointer reverts to its normal state after the script finishes executing.

1 It is good programming practice to use an ENDWAITCURSOR statement with every BEGINWAITCURSOR statement.

Example

```
BEGINWAITCURSOR
0   WAIT FOR 30
1   ENDWAITCURSOR
```

In the above example, the pointer is set to Busy, script execution is paused for 30 seconds, and then the pointer reverts to its normal state.

{button ,AL(`CLS_CorelScriptTools;FNC_EndWaitCursor')} **Related Topics**

CorelScriptTools.BuildDate

Function **BuildDate**(ByVal **Year** AS Long, ByVal **Month** AS Long, ByVal **Day** AS Long) AS Date

CorelScriptTools

Description

Assigns a date value to a date variable.

0 BUILDATE can only accept a date value between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050). If a date outside this range is specified, an error occurs.

Parameters		Description
Year		Numeric expression specifying the year to assign to a date variable.
0	**Month**	Numeric expression specifying the month to assign to a date variable. Valid values are from 1 to 12 inclusive.
1	**Day**	Numeric expression specifying the day to assign to a date variable. Valid values are from 1 to 31 inclusive, depending on the Month setting.

Example

```
DIM BigDay AS DATE
BigDay = BUILDDATE(1996, 1, 14)
```

In the above example, the first line declares the date variable BigDay. This variable is then assigned the date January 14, 1996.

{button ,AL(`CLS_CorelScriptTools;FNC_BuildDate')} **Related Topics**

CorelScriptTools.BuildTime

Function **BuildTime**(ByVal **Hour** AS Long, ByVal **Minute** AS Long, ByVal **Second** AS Long) AS Date

CorelScriptTools

Description

Assigns a time value to a date variable.

Parameters	Description
Hour	Numeric expression specifying the hour to assign to a date variable. This value is based on a 24-hour clock. For example, 5 PM equals 17. Valid values are from 0 to 23 inclusive.
0	**Minute** Numeric expression specifying the minute to assign to a date variable. Valid values are from 0 to 59 inclusive.
1	**Second** Numeric expression specifying the second to assign to a date variable. Valid values are from 0 to 59 inclusive.

Example

```
DIM BigDay AS DATE
BigDay = BUILDTIME(14, 30, 0)
```

In the above example, the first line declares the date variable BigDay. This variable is then assigned the time 2:30 PM.

{button ,AL(`CLS_CorelScriptTools;FNC_BuildTime')} **Related Topics**

CorelScriptTools.Dec

Function **Dec**(ByVal **Hex** AS String) AS Long

CorelScriptTools

Description

Returns the conversion of a hexadecimal value into decimal notation (as a long data type).

Parameter	Description
x	A string expression representing a hexadecimal number.

Notes

- Decimal notation is a numerical system based on groups of ten units.
- The highest value you can convert is 7FFFFFFF.
- The HEX function performs the opposite conversion, from decimal to hexadecimal.

Examples

```
x& = DEC("A27")
```

In the above example, **x** equals 2599.

{button ,AL(`CLS_CorelScriptTools;FNC_Dec')} **Related Topics**

CorelScriptTools.FileAttr

Function **FileAttr**(ByVal **FolderFile** AS String) AS Long

CorelScriptTools

Description

This functions returns a file's or folder's attributes.

Parameters	Description
FolderFile	String expression specifying the file/folder for which the attributes are returned.

0 Example

```
retval = FILEATTR("C:\myfiles\mysetup.txt")
```

If MYSETUP.TXT is read-only, hidden, and a system file, retval equals 7.

0 In cases where multiple attributes are returned, you can use the AND (bitwise) operator to determine specific attributes. To determine if MYSETUP.TXT was a read-only file you could use the following syntax:

```
IF 1 AND retval THEN readOnly$ = "Yes" ELSE readOnly$ = "No"
```

1 is the read-only attribute. The variable readOnly is assigned a string based on bitwise comparison. In this case readOnly is assigned "Yes".

{button ,AL(`CLS_CorelScriptTools;FNC_FileAttr')} **Related Topics**

CorelScriptTools.FindFirstFolder

Function **FindFirstFolder**(ByVal **SearchCriteria** AS String, ByVal **Attributes** AS Long) AS String

CorelScriptTools

Description

Use the FINDFIRSTFOLDER and FINDNEXTFOLDER functions to assemble or perform an operation on a list of files, folders, or both. The FINDFIRSTFOLDER function is used to locate the first file or first folder in a folder that meets a specified search criteria. The FINDNEXTFOLDER function is used to locate the next file or next folder that meets the specified search criteria set by the FINDFIRSTFOLDER. The FINDNEXTFOLDER function must be used in conjunction with the FINDFIRSTFOLDER function.

0 Specifying folder in the attributes parameter (16) by itself does not specify a type of folder. You must use another parameter along with 16 to specify a folder type.

1 Example

```
DIM DCOUNT%,  FCOUNT%             'creates 2 integer variables
DIM FILESARR$(100), DIRARR$(100)  'creates 2 string arrays

0   REM LOOP #1
1   REM FIND ALL DIRECTORIES IN THE SAMPLES FOLDER
2   DCOUNT = 1
3   DIRARR(DCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\*", 16 OR 128)
4   WHILE (DIRARR(DCOUNT) <> "")
    5   MESSAGE DIRARR(DCOUNT)
    6   IF DIRARR(DCOUNT) <> "." AND DIRARR(DCOUNT) <> ".." THEN DCOUNT = DCOUNT + 1
    7   DIRARR(DCOUNT) = FINDNEXTFOLDER()
8   WEND

9   REM LOOP #2

10  REM FIND ALL *.VP FILES IN EACH DIRECTORY FOUND IN EARLIER LOOP
11  DIM I%
12  FCOUNT = 1
13  FOR I% = 1 TO DCOUNT-1
    14  FILESARR(FCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\" + DIRARR(I%) + "\*.VP", 1
OR 2 OR 4 OR 32 OR 128)
    15  WHILE (FILESARR(FCOUNT) <> "" )
    16  MESSAGE DIRARR(I%) & CHR(13) & FILESARR(FCOUNT)
        17      FCOUNT = FCOUNT + 1
        18      FILESARR(FCOUNT) = FINDNEXTFOLDER()
    19  WEND
20  NEXT I%
```

0 In the above example, the first loop fills an array (DIRARR) with the names of the normal folders in the D:\COREL\VENTURA\SAMPLES folder. The second loop searches the folders in the SAMPLES folder for any file with the extension VP. Any found VP file has its name added to the FILESARR array and has its name displayed in a message box.

1 The following statement in the first loop is used to remove the current (.) folder and parent (..) folder from being sent to the DIRARR array:

```
IF DirArr(Dcount) <> "." AND DirArr(Dcount) <> ".." THEN Dcount = Dcount + 1
```

{button ,AL(`CLS_CorelScriptTools;FNC_FindFirstFolder')} **Related Topics**

CorelScriptTools.FindNextFolder

Function **FindNextFolder**() AS String

CorelScriptTools

Description

Use the FINDFIRSTFOLDER and FINDNEXTFOLDER functions to assemble or perform an operation on a list of files, folders, or both. The FINDFIRSTFOLDER function is used to locate the first file or first folder in a folder that meets a specified search criteria. The FINDNEXTFOLDER function is used to locate the next file or next folder that meets the specified search criteria set by the FINDFIRSTFOLDER. The FINDNEXTFOLDER function must be used in conjunction with the FINDFIRSTFOLDER function.

0 Specifying folder in the attributes parameter (16) by itself does not specify a type of folder. You must use another parameter along with 16 to specify a folder type.

1 Example

```
DIM DCOUNT%,  FCOUNT%             'creates 2 integer variables
DIM FILESARR$(100), DIRARR$(100)  'creates 2 string arrays

0   REM LOOP #1
1   REM FIND ALL DIRECTORIES IN THE SAMPLES FOLDER
2   DCOUNT = 1
3   DIRARR(DCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\*", 16 OR 128)
4   WHILE (DIRARR(DCOUNT) <> "")
    5   MESSAGE DIRARR(DCOUNT)
    6   IF DIRARR(DCOUNT) <> "." AND DIRARR(DCOUNT) <> ".." THEN DCOUNT = DCOUNT + 1
    7   DIRARR(DCOUNT) = FINDNEXTFOLDER()
8   WEND

9   REM LOOP #2

10  REM FIND ALL *.VP FILES IN EACH DIRECTORY FOUND IN EARLIER LOOP
11  DIM I%
12  FCOUNT = 1
13  FOR I% = 1 TO DCOUNT-1
    14  FILESARR(FCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\" + DIRARR(I%) + "\*.VP", 1
OR 2 OR 4 OR 32 OR 128)
    15  WHILE (FILESARR(FCOUNT) <> "" )
    16  MESSAGE DIRARR(I%) & CHR(13) & FILESARR(FCOUNT)
        17      FCOUNT = FCOUNT + 1
        18      FILESARR(FCOUNT) = FINDNEXTFOLDER()
    19  WEND
20  NEXT I%
```

0 In the above example, the first loop fills an array (DIRARR) with the names of the normal folders in the D:\COREL\VENTURA\SAMPLES folder. The second loop searches the folders in the SAMPLES folder for any file with the extension VP. Any found VP file has its name added to the FILESARR array and has its name displayed in a message box.

1 The following statement in the first loop is used to remove the current (.) folder and parent (..) folder from being sent to the DIRARR array:

```
IF DirArr(Dcount) <> "." AND DirArr(Dcount) <> ".." THEN Dcount = Dcount + 1
```

{button ,AL(`CLS_CorelScriptTools;FNC_FindNextFolder')} **Related Topics**

CorelScriptTools.FormatTime

Function **FormatTime**(ByVal **Time** AS Date, ByVal **Format** AS String) AS String

CorelScriptTools

Description

Converts a time expression into a string with a specified format.

Return Value

Lets you specify the string variable that is assigned the formatted time.

0	Hours as 1-12 (12-hour clock)	h
1	Hours as 01-12 (12-hour clock)	hh
2	Hours as 0-23 (24-hour clock)	H
3	Hours as 00-23 (24-hour clock)	HH
4	Minutes as 0-59	m
5	Minutes as 00-59	mm
6	Seconds as 0-59	s
7	Seconds as 00-59	ss
8	AM/PM as A or P	t
9	AM/PM as AM or PM	tt
10	Time as 4:36 pm	h:mm pm
11	You can insert spaces and punctuation between time elements within the formatting string. See the example below.	

Parameters	Description
Time	Lets you specify the time expression to convert to a string.
0	**Format** Lets you specify a code, as a string, representing the format of the time. Formats are created by using and combining the following codes.

0 Example

```
DIM TimeNow AS DATE
TimeNow = GETCURRDATE()
0   StringTime = FORMATTIME (TimeNow, "HH:mm:ss tt")
1   MESSAGE StringTime
```

In the above example, the first line declares the date variable TodayDate. This variable is then assigned the current date and time with the GETCURRDATE function. The StringTime variable is then assigned the current time using the formatting shown in the following example.

0 13:46:25 PM

{button ,AL(`CLS_CorelScriptTools;FNC_FormatTime')} **Related Topics**

CorelScriptTools.FromCentimeters

Function **FromCentimeters**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from centimeters to tenths of a micron.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMCENTIMETERS(8), FROMCENTIMETERS(-5), FROMCENTIMETERS(0),
FROMCENTIMETERS(2.5), FROMCENTIMETERS(0.75)
```

This CorelDRAW command would create a rectangle 7.5 by 8 centimeters. The rectangle's top left corner coordinate is -5, 8 centimeters relative to the center of the page, and the corners are 0.75 centimeters in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromCentimeters')} **Related Topics**

CorelScriptTools.FromCiceros

Function **FromCiceros**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from ciceros to tenths of a micron.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMCICEROS(18), FROMCICEROS(-12), FROMCICEROS(8), FROMCICEROS(6),
FROMCICEROS(1.5)
```

This CorelDRAW command would create a rectangle 18 by 10 ciceros. The rectangle's top left corner coordinate is -12, 18 ciceros relative to the center of the page, and the corners are 1.5 ciceros in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromCiceros')} **Related Topics**

CorelScriptTools.FromDidots

Function **FromDidots**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from didots to tenths of a micron.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMDIDOTS(50), FROMDIDOTS(-70), FROMDIDOTS(0), FROMDIDOTS(30), FROMDIDOTS(20)
```

This CorelDRAW command would create a rectangle 100 by 50 didots. The rectangle's top left coordinate is -70, 50 didots relative to the center of the page, and the corners are 20 didots in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromDidots')} **Related Topics**

CorelScriptTools.FromInches

Function **FromInches**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from inches to tenths of a micron.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMINCHES(3), FROMINCHES(-2), FROMINCHES(0), FROMINCHES(1), FROMINCHES(0.25)
```

This CorelDRAW command would create a rectangle 3 by 3 inches. The rectangle's top left coordinate is -2, 3 inches relative to the center of the page, and the corners are 0.25 inches in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromInches')} **Related Topics**

CorelScriptTools.FromPicas

Function **FromPicas**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from picas to tenths of a micron.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMPICAS(18), FROMPICAS(-12), FROMPICAS(8), FROMPICAS(6), FROMPICAS(2)
```

This CorelDRAW command would create a rectangle 18 by 10 picas. The rectangle's top left coordinate is -12, 18 picas relative to the center of the page, and the corners are 2 picas in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromPicas')} **Related Topics**

CorelScriptTools.FromPoints

Function **FromPoints**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from points to tenths of a micron

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.CreateRectangle FROMPOINTS(210), FROMPOINTS(-140), FROMPOINTS(90), FROMPOINTS(70),
FROMPOINTS(1.75)
```

This CorelDRAW command would create a rectangle 210 by 140 points. The rectangle's top left corner coordinate is -140, 210 points relative to the center of the page, and the corners are 1.75 points in diameter.

{button ,AL(`CLS_CorelScriptTools;FNC_FromPoints')} **Related Topics**

CorelScriptTools.GetAppHandle

Function **GetAppHandle**() AS Long

CorelScriptTools

Description

Returns the Application Instance Handle for the Corel application that is running a script. For example, if you are running the script from the Corel SCRIPT Editor, GETAPPHANDLE returns the Editor's Application Instance Handle. If you run a script from Corel VENTURA, GETAPPHANDLE returns VENTURA's Application Instance Handle. This function is used in conjunction with DLL calls that require the application's handle.

Return Value

Lets you specify a numeric variable that is passed the Application Instance Handle.

Example

```
hand = GETAPPHANDLE()
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetAppHandle')} **Related Topics**

CorelScriptTools.GetColor

Function **GetColor**(ByRef **Red** AS Long, ByRef **Green** AS Long, ByRef **Blue** AS Long) AS Boolean

CorelScriptTools

Description

Displays a standard Windows Color dialog box and returns color setting values from the RGB color model (Red, Green, Blue).

Parameters	Description
Red	Lets you specify the numeric variable that is passed the Red setting of the selected color (0 - 255). You can also use this variable to set an initial value.
0	**Green** Lets you specify the numeric variable that is passed the Green setting of the selected color (0 - 255). You can also use this variable to set an initial value.
1	**Blue** Lets you specify the numeric variable that is passed the Blue setting of the selected color (0 - 255). You can also use this variable to set an initial value.

Example

```
GETCOLOR MyRed%, MyGreen%, MyBlue%
```

The above example displays the following dialog box and returns the RGB color settings for the selected color to the numeric variables MyRed, MyGreen, and MyBlue.

{button ,AL(`CLS_CorelScriptTools;FNC_GetColor')} **Related Topics**

CorelScriptTools.GetCommandLine

Function **GetCommandLine**() AS String

CorelScriptTools

Description

Returns the parameters used in the command line that launch a script. To test this command, specify a command line in the Command Line text box in the Corel SCRIPT Editor's Options dialog box (click Tools, Options, Environment tab). For more information about command lines, click .

Return Value

String variable that is passed the command line parameters.

Example

```
CommandLine$ = GETCOMMANDLINE ( )
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetCommandLine')} **Related Topics**

CorelScriptTools.GetCurrFolder

Function **GetCurrFolder**() AS String

CorelScriptTools

Description

Returns the name of the active Windows folder and path.

Note

You can set the active folder using the SETCURRFOLDER statement.

0 In Corel SCRIPT version 7.0, the GETCURRFOLDER function and the SETCURRFOLDER statement replace the CURRFOLDER statement.

1 Example

```
SETCURRFOLDER "c:\corel\graphics8\scripts\"
folder$ = GETCURRFOLDER ( )
0   MyFile$ = "\MyScript.csc"
1   MyPathFile = folder$ & MyFile&
```

0 In the above example the first line sets the active folder. The second line assigns the active folder to a string variable. The third line assigns a file name to a string variable. In the last line a string variable is assigned a value which is made by combining the folder and file string variables.

{button ,AL(`CLS_CorelScriptTools;FNC_GetCurrFolder')} **Related Topics**

CorelScriptTools.GetDateInfo

Sub **GetDateInfo**(ByVal **Date** AS Date, ByRef **Year** AS Long, ByRef **Month** AS Long, ByRef **Day** AS Long, ByRef **DayOfWeek** AS Long)

CorelScriptTools

Description

Extracts the components of a date expression to numeric variables.

Parameters	Description
Date	Lets you specify the date expression to extract components from.
0	**Year** Lets you specify the numeric variable that is assigned the year component from the specified date expression.
1	**Month** Lets you specify the numeric variable that is assigned the month component from the specified date expression.
2	**Day** Lets you specify the numeric variable that is assigned the day component from the specified date expression.
3	**DayOfWeek** Lets you specify the numeric variable that is assigned the day of week component from the specified date expression. Sunday corresponds to 1, Monday to 2, and so on.

GETDATEINFO can only accept a date value between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050). If a date outside this range is specified, an error occurs.

Example

```
DIM TodayDate AS DATE
TodayDate = GETCURRDATE()
0   GETDATEINFO TodayDate, Y&, M&, D&, DW&
```

In the above example, the first line declares the date variable TodayDate. This variable is then assigned the current date with the GETCURRDATE function. The variables Y, M, D, and DW are then assigned their respective component of the date stored in TodayDate. If TodayDate was set to May 29, 1996, then Y=1996, M=5, D=29, and DW=4.

{button ,AL(`CLS_CorelScriptTools;FNC_GetDateInfo')} **Related Topics**

CorelScriptTools.GetFileBox

Function **GetFileBox**(ByVal Filter AS String, ByVal Title AS String, ByVal Type AS Long, ByVal File AS String, ByVal Extension AS String, ByVal Folder AS String, ByVal Button AS String) AS String

CorelScriptTools

Description

This function displays a standard Windows File Open or File Save As dialog box. Both dialog boxes allow users to choose a file from the file system. The GETFILEBOX function returns the selected filename and its full path, or an empty string if the user chooses Cancel. The GETFILEBOX statement by itself does not open or save a file; it only returns a string corresponding to the selected file.

Parameters	Description
Filter	String expression specifying the filters to use in the dialog box. For the Open dialog box, the filters are listed in the Files of Type list box. For the Save As dialog box, the filters are listed in the Save as Type list box. Filters are specified in two parts. The first part is the text that appears in the list box, and the second part is the actual filter extension. The parts are separated by the | character (do not use spaces before or after the | characters). To separate multiple filters, use the | character. See the example below for more information.
0	**Title** String expression specifying the title to display in the dialog box. If not specified, "Open" is displayed for an Open dialog box and "Save As" is displayed for a Save As dialog box.
1	**Type** Numeric expression specifying the type of dialog box to display: FileOpen dialog box (0) or FileSave dialog box (1) - default is 0.
2	**File** String expression specifying the text to display in the File name text box of the dialog box. If not specified, the text box is empty.
3	**Extension** String expression specifying the default extension to append to a File name if the user omits the extension.
4	**Folder** String expression specifying the default folder used by the dialog box. If not specified, or the specified folder does not exist, the current folder is used.
5	**Button** String expression specifying a button name to override the Open or Save button in the dialog box. If not specified, the button's name remains unchanged.

0 Example

```
SETCURRFOLDER = "c:\COREL50\DRAW\samples" 'set the current folder
Filename$=GETFILEBOX("Included Scripts|*.csc|All Files|*.*", "Scripts included...", 0,"animals")
```

Displays an Open dialog box.

{button ,AL(`CLS_CorelScriptTools;FNC_GetFileBox')} **Related Topics**

CorelScriptTools.GetFolder

Function **GetFolder**(ByVal InitFolder AS String) AS String

CorelScriptTools

Description

This function displays a Windows Choose Folder dialog box. The Choose Folder dialog box returns the folder and path a user chooses as a string.

0 If the Cancel button is clicked, an empty string is returned.

Parameters	Description
InitFolder	String expression specifying the default path and folder to display in the dialog box. If not specified, the active folder is used.

0 Example

```
NewFolder$ = GETFOLDER("D:\Corel60")
SETCURRFOLDER = NewFolder$
```

The selected folder is passed to the string variable NewFolder. The SETCURRFOLDER statement sets the current folder to the folder name passed to NewFolder.

{button ,AL(`CLS_CorelScriptTools;FNC_GetFolder')} **Related Topics**

CorelScriptTools.GetFont

Function **GetFont**(ByRef **FaceName** AS String, ByRef **PointSize** AS Long, ByRef **Weight** AS Long, ByRef **Italic** AS Boolean, ByRef **Underline** AS Boolean, ByRef **StrikeOut** AS Boolean, ByRef **Red** AS Long, ByRef **Green** AS Long, ByRef **Blue** AS Long) AS Boolean

CorelScriptTools

Description

Displays a standard Windows Font dialog box and returns the selected font settings. Most Windows fonts only use two weight settings: 400 (Normal) and 700 (Bold).

Parameters	Description
FaceName	Lets you specify a string variable that is passed the name of the selected font. You can also use this variable to set an initial value.
0	**PointSize** Lets you specify a numeric variable that is passed the font size in points. You can also use this variable to set an initial value. This parameter uses non-fractional values
1	**Weight** Lets you specify a numeric variable that is passed the font's weight setting (number of inked pixels per 1000 pixels). Common values and their corresponding names include: 100 Thin 0 200 Extra Light, Ultra Light 1 300 Light 2 400 Normal, Regular 3 500 Medium 4 600 Semi Bold, Demi Bold 5 700 Bold 6 800 Extra Bold, Ultra Bold 7 900 Black, Heavy
Italic	Lets you specify a numeric variable that is passed the font's italic setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
0	**Underline** Lets you specify a numeric variable that is passed the font's underline setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
1	**StrikeOut** Lets you specify a numeric variable that is passed the font's strike out setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
2	**Red** Lets you specify the numeric variable that is passed the Red (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.
3	**Green** Lets you specify the numeric variable that is passed the Green (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.
4	**Blue** Lets you specify the numeric variable that is passed the Blue (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.

0 Example

```
GETFONT FN, PS, Wt, Italic, UL, SO, R, G, B
```

The above example displays the Font dialog box and returns the font settings the variables specified.

{button ,AL(`CLS_CorelScriptTools;FNC_GetFont')} **Related Topics**

CorelScriptTools.GetProcessInfo

Function **GetProcessInfo**(ByVal **ProcessHandle** AS Long) AS Long

CorelScriptTools

Description

This function returns the status of an executable.

0 Numeric variable that is passed a value that indicates whether an executable is running. If the executable is running, this variable is passed the value 259.

Parameters	Description
ProcessHandle	Numeric expression specifying the Windows Process Handle of an executable. Use the STARTPROCESS function to determine an executable's Windows Process Handle.

0 Example

```
launch = STARTPROCESS ("C:\WINDOWS\CALC.EXE")
... 'other script statements
0   ... 'other script statements
1   ... 'other script statements
2   Calc_Status = GETPROCESSINFO (launch)
```

The above example launches the Windows Calculator and passes the Windows Process Handle to the launch variable. The launch variable is used with the GETPROCESSINFO function to determine whether the Calculator is running.

{button ,AL(`CLS_CorelScriptTools;FNC_GetProcessInfo')} **Related Topics**

CorelScriptTools.GetScriptFolder

Function **GetScriptFolder**() AS String

CorelScriptTools

Description

Returns the path and the folder where the executing script resides. If this function is used in a Corel SCRIPT Executable, it returns the path and folder where the Executable resides.

0 If the script has not been previously saved to disk or a network, this function returns an empty string.

1 Example

```
folder = GETSCRIPTFOLDER ( )
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetScriptFolder')} **Related Topics**

CorelScriptTools.GetTempFolder

Function **GetTempFolder**() AS String

CorelScriptTools

Description

Returns the path and the folder of the system's Windows temporary folder.

0 Example

```
t_folder = GETTEMPFOLDER ( )
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetTempFolder')} **Related Topics**

CorelScriptTools.GetTimeInfo

Sub **GetTimeInfo**(ByVal **Time** AS Date, ByRef **Hour** AS Long, ByRef **Minute** AS Long, ByRef **Second** AS Long)

CorelScriptTools

Description

Extracts the components of a time expression to numeric variables.

Parameters	Description
Time	Lets you specify the time expression to extract components from.
0	**Hour** Lets you specify the numeric variable that is assigned the hour component from the specified time expression. The number assigned is based on a 24-hour clock. For example, 16 is the numeric variable for 4pm.
1	**Minute** Lets you specify the numeric variable that is assigned the minute component from the specified time expression.
2	**Second** Lets you specify the numeric variable that is assigned the second component from the specified time expression.

3 **Example**

```
DIM TodayTime AS DATE
TodayTime = GETCURRDATE()
0   GETTIMEINFO TodayTime, H&, M&, S&
```

In the above example, the first line declares the date variable TodayTime. This variable is then assigned the current date and time with the GETCURRDATE function. The variables H, M, and S are then assigned their respective component of the time stored in TodayTime. If TodayTime was set to 5:37:16 PM, then H=17, M=37, S=16.

{button ,AL(`CLS_CorelScriptTools;FNC_GetTimeInfo')} **Related Topics**

CorelScriptTools.GetType

Function **GetType**(ByVal **Expression** AS Variant) AS Long

CorelScriptTools

Description

Returns an expression's data type. In the case of variants, the data subtype is returned. See Variants for more information.

Parameters	Description
Expression	Lets you specify the expression to use.

Example

```
x% = 66    'integer data type
y# = 14    'long data type
0   Z = x / y
1   A = GETTYPE (x / y)
```

In the above example, A is assigned the value 2 since the variant Z data subtype is set to long. In the following example, B is set to 4 since a whole number is treated as a long and C is set to 6 since a fractional number is treated as a double.

```
B = GETTYPE (3)
C = GETTYPE (3.3)
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetType')} **Related Topics**

CorelScriptTools.GetVersion

Function **GetVersion**(ByVal **Option** AS Long) AS Long

CorelScriptTools

Description

Returns the system or Corel SCRIPT version numbers.

0 For a script, Executable, DLL, or Corel Add-on created with Corel SCRIPT to run, the major version numbers for Corel SCRIPT (the compiler) and the Corel SCRIPT run-time interpreter (SCINTxx.DLL) must be the same or else an error will occur. A difference in the minor version numbers will not cause an error.

Parameters	Description
Option	Lets you specify the system or Corel SCRIPT component to query: 0 0 Corel SCRIPT run-time interpreter version (the SCINTxx.DLL file being used with the current session of Corel SCRIPT) 1 10 Corel SCRIPT compiler version number 2 30 Windows platform 3 31 Windows major version number 4 32 Windows minor version number 5 33 Windows build number

0 Example

```
CS_version = GETVERSION(10)
MESSAGE "Corel SCRIPT major version number " & LEFT (CS_version, 2)
0   MESSAGE "Corel SCRIPT minor version number " & RIGHT (CS_version, 2)
```

In the above example, the first line passes the Corel SCRIPT version number to CS_version. A message box is then used to display the first two digits in CS_version using the LEFT function. Next, a message box is used to display the first two digits in CS_version using the RIGHT function.

{button ,AL(`CLS_CorelScriptTools;FNC_GetVersion')} **Related Topics**

CorelScriptTools.GetWinHandle

Function **GetWinHandle**() AS Long

CorelScriptTools

Description

Returns the window handle for the window that is running the script. For example, if you are running the script from the Corel SCRIPT Editor, GETWINHANDLE returns the Editor's Windows handle. If you run a script from CorelDRAW, GETWINHANDLE returns DRAW's Windows handle. This function is used in conjunction with DLL calls that require the window's handle.

0 Example

```
hand = GETWINHANDLE()
```

{button ,AL(`CLS_CorelScriptTools;FNC_GetWinHandle')} **Related Topics**

CorelScriptTools.Kill

Function **Kill**(ByVal **FileName** AS String) AS Boolean

CorelScriptTools

Description

Deletes a file. This statement is the same as clicking File, Delete in the Windows Explorer or in My Computer in Windows 95.

Parameters	Description
FileName	String expression specifying the filename to delete. You can use wild cards (* and ?) if you want to delete a group of files. For example, script*.* deletes all the files in the current folder beginning with script. Using script?.* deletes all the files in the current folder that begin with script and are followed by only one more character.

0 Example

```
KILL "temp.out"
```

Deletes the file TEMP.OUT in the current folder.

```
KILL "C:\MyDocs\temp.out"
```

Deletes the file TEMP.OUT in the C:\MyDocs folder.

{button ,AL(`CLS_CorelScriptTools;FNC_Kill')} **Related Topics**

CorelScriptTools.LengthConvert

Function **LengthConvert**(ByVal **FromUnit** AS Long, ByVal **ToUnit** AS Long, ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a number from one length measurement to another.

Parameter	Description
x	Any number from 1 to 7 that indicates the unit of measurement from which to convert. 1 inches 0 2 centimeters 1 3 points 2 4 Ciceros 3 5 didots 4 6 picas 5 7 CorelDRAW and VENTURA units (tenths of a micron
y	Any number from 1 to 7 that indicates the unit of measurement to convert to: 1 inches 0 2 centimeters 1 3 points 2 4 Ciceros 3 5 didots 4 6 picas 5 7 CorelDRAW and VENTURA units (tenths of a micron)
z	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
x_microns = LENGTHCONVERT(1, 7, 1)
```

The above example converts one inch to tenths of a micron. The variable **x_microns** equals 254,000.

{button ,AL(`CLS_CorelScriptTools;FNC_LengthConvert')} **Related Topics**

CorelScriptTools.Log

Function **Log**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Returns the base-10 logarithm of a number. The LOG function uses a base of 10. If another base is needed, use LOG(x)/LOG(b) formula where b is the base.

Parameters	Description
Value	Any positive numeric expression.

0 Example

```
x = LOG(25)
y = LOG(5)
```

In the above example, x equals 1.397940 and y equals 0.6989700.

{button ,AL(`CLS_CorelScriptTools;FNC_Log')} **Related Topics**

CorelScriptTools.MkFolder

Function **MkFolder**(ByVal **Folder** AS String) AS Boolean

CorelScriptTools

Description

Creates a new folder.

Parameters	Description
Folder	String expression specifying the name of the folder to be created. Path information is optional.

0 Example

```
MKFOLDER "work"
```

Creates the folder work as a subfolder of the current folder.

```
success = MKFOLDER ("work")
```

Creates the folder work as a subfolder of the current folder and assigns -1 to success.

{button ,AL(`CLS_CorelScriptTools;FNC_MkFolder')} **Related Topics**

CorelScriptTools.RegistryQuery

Function **RegistryQuery**(ByVal **MainKey** AS Long, ByVal **SubKey** AS String, ByVal **Value** AS String) AS Variant

CorelScriptTools

Description

Returns the value data of a specified value key in the system's Windows registry. This function can help you determine where programs and files are installed on a user's system. This type of information is important when creating scripts that are to run on different system setups.

0 Lets you specify the variable that is passed the value data of a specified value key in the Windows registry. Since this function can pass a string or numeric value, the variable you specify should be a variant. You can use the GETTYPE function to determine a variant's subtype.

Parameters	Description
MainKey	Lets you specify the main registry value key to query: 0 0 HKEY_CLASSES_ROOT 1 1 HKEY_CURRENT_USER 2 2 HKEY_LOCAL_MACHINE 3 3 HKEY_USERS 4 4 HKEY_PERFORMANCE_DATA 5 5 HKEY_CURRENT_CONFIG 6 6 HKEY_DYN_DATA
7	**SubKey** Lets you specify the sub registry value key to query. This must be a complete key path. In Windows 95 for example, "SOFTWARE\Microsoft\Windows\CurrentVersion\Fonts" is a complete path.
8	**Value** Lets you specify the registry value key to query. To specify a default value, use an empty string. Specify an empty string by using two quotation marks ("").

0 Example

```
Config_Ventura = REGISTRYQUERY (2, "SOFTWARE\Corel\Corel Ventura\7.0", "ConfigDir")
```

The above example returns the root folder where Corel VENTURA 7 is installed.

```
Arial_file = REGISTRYQUERY (2, "SOFTWARE\Microsoft\Windows\CurrentVersion\Fonts", "Arial
(TrueType)")
```

The above example returns the Arial True Type font's file name.

```
YourName$ = REGISTRYQUERY(2,"SOFTWARE\Corel","UserName")
```

The above example returns the name of the registered owner of Corel Software.

```
CompanyName$ = REGISTRYQUERY(2,"SOFTWARE\Corel","ORGANIZATION")
```

The above example returns the organization name of the registered owner of Corel Software.

```
Phone$ = REGISTRYQUERY(2,"SOFTWARE\Corel","PHONENUMBER")
```

The above example returns the phone number of the registered owner of Corel Software

---

{button ,AL(`CLS_CorelScriptTools;FNC_RegistryQuery')} **Related Topics**

CorelScriptTools.Rename

Function **Rename**(ByVal **Src** AS String, ByVal **Dst** AS String, ByVal Overwrite AS Long) AS Boolean

CorelScriptTools

Description

The RENAME statement changes the name of a file or folder, or can be used to move a file. You cannot move a folder using the RENAME statement.

0 You can also use RENAME as a function: it returns TRUE (-1) if the RENAME operation is successful, FALSE (0) if is not.

Parameters	Description
Src	String expression specifying the name of the file or folder to move. file_folder1 can include drive and folder path specifics if path specifics are not included, RENAME assumes the current folder.
0	**Dst** String expression specifying the name of the file where file_folder1 is to be moved. file_folder2 can include drive and folder path specifics if path specifics are not included, RENAME assumes the current folder.
1	**Overwrite** If file_folder2 already exists, determines whether to overwrite the existing file (you cannot overwrite existing folders):
	2 0 = rename and overwrite
	3 1 = overwrite fails (default if omitted)

0 Example

```
' statement example
DIM x AS STRING
0   DIM y AS STRING
1   x = "C:\work\example1.vp"
2   y = "D:\work\example1.vp"
3   RENAME x, y, 0
```

The above example moves the EXAMPLE1.VP file to the Work folder on the D drive.

0 ' statement example

```
DIM x AS STRING
0   DIM y AS STRING
1   x = "C:\work\example1.cdr"
2   y = "C:\work\example2.cdr"
3   success = RENAME (x, y, 0)
```

The above example renames the EXAMPLE1.CDR file to EXAMPLE2.CDR, and assigns -1 to success.

{button ,AL(`CLS_CorelScriptTools;FNC_Rename')} **Related Topics**

CorelScriptTools.RmFolder

Function **RmFolder**(ByVal **Folder** AS String) AS Boolean

CorelScriptTools

Description

Removes an existing folder. The folder must be empty before it can be deleted. You can also use RMFOLDER as a function: it returns TRUE (-1) if the folder was removed, FALSE (0) if it was not.

Parameters	Description
Folder	String expression specifying the name of the folder to remove. folderName can include drive specifics if drive specifics are not included, RENAME assumes the current drive.

0 Example

```
RMFOLDER "C:\TEMP\WORK"
```

Removes the Work folder from the Temp folder.

```
success% = RMFOLDER "C:\TEMP\WORK"
```

Removes the Work folder from the Temp folder and assigns -1 to success.

{button ,AL(`CLS_CorelScriptTools;FNC_RmFolder')} **Related Topics**

CorelScriptTools.ToCentimeters

Function **ToCentimeters**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

0 Converts a numeric value from tenths of a micron to centimeters.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
0   xCM = TOCENTIMETERS (x)
1   yCM = TOCENTIMETERS (y)
```

In this CorelDRAW example, **xCM** and **yCM** are set to the X and Y coordinates of the selected object in centimeters.

{button ,AL(`CLS_CorelScriptTools;FNC_ToCentimeters')} **Related Topics**

CorelScriptTools.ToCiceros

Function **ToCiceros**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

0 Converts a numeric value from tenths of a micron to ciceros.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
0   xCiceros = TOCICEROS (x)
1   yCiceros = TOCICEROS (y)
```

In this CorelDRAW example, **xCiceros** and **yCiceros** are set to the X and Y coordinates of the selected object in ciceros.

{button ,AL(`CLS_CorelScriptTools;FNC_ToCiceros')} **Related Topics**

CorelScriptTools.ToDidots

Function **ToDidots**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from tenths of a micron to didots.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
0   xDidots = TODIDOTS (x)
1   yDidots = TODIDOTS (y)
```

In this CorelDRAW example, **xDidots** and **yDidots** are set to the X and Y coordinates of the selected object in didots.

{button ,AL(`CLS_CorelScriptTools;FNC_ToDidots')} **Related Topics**

CorelScriptTools.ToInches

Function **ToInches**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from tenths of a micron to inches.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
0   xInch = TOINCHES (x)
1   yInch = TOINCHES (y)
```

In this CorelDRAW example, **xInch** and **yInch** are set to the X and Y coordinates of the selected object in inches.

{button ,AL(`CLS_CorelScriptTools;FNC_ToInches')} **Related Topics**

CorelScriptTools.ToPicas

Function **ToPicas**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

Converts a numeric value from tenths of a micron to picas.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
0   xPica = TOPICAS (x)
1   yPica = TOPICAS (y)
```

In this CorelDRAW example, **xPica** and **yPica** are set to the X and Y coordinates of the selected object in picas.

{button ,AL(`CLS_CorelScriptTools;FNC_ToPicas')} **Related Topics**

CorelScriptTools.ToPoints

Function **ToPoints**(ByVal **Value** AS Double) AS Double

CorelScriptTools

Description

0 Converts a numeric value from tenths of a micron to points.

Parameter	Description
x	Any numeric expressions specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

```
.GetPosition (x,y)
xPoint = TOPOINTS (x)
yPoint = TOPOINTS (y)
```

In this CorelDRAW example, **xPoint** and **yPoint** are set to the X and Y coordinates of the selected object in points.

{button ,AL(`CLS_CorelScriptTools;FNC_ToPoints')} **Related Topics**

StructFontProperties properties

StructFontProperties Legend

- Fill
- Name
- Outline
- Overscore
- Position
- RangeKerning
- Size
- Strikethru
- Style
- Underline
- Uppercase

StructFontProperties

Class **StructFontProperties**

Properties Referenced by

The **StructFontProperties** class defines font objects in CorelDRAW and describes the look and behavior of the object through its properties. For example, a font character can be uppercase, underlined, outlined, filled, or resized.

In CorelDRAW, you can create and edit custom characters. The unique design of a character set is called its typeface. A font is a complete set of characters that share a common typeface -uppercase and lowercase letters, numbers, punctuation marks, and symbols.

A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font. The font list is a collection of font styles in CorelDRAW.

Before you can use a new or modified typeface, you must install it. You can install TrueType fonts using the Windows Control Panel or the Font Navigator Utility. To install Type 1 fonts, you must use the Adobe Type Manager. For information about installing Adobe Type 1 fonts, refer to the documentation provided with Adobe Type Manager.

If someone else plans to view your file, each font you use in your document must also be installed on their machine. Otherwise, CorelDRAW substitutes the font using PANOSE. (For more information about PANOSE, see "Substituting unavailable fonts." To avoid this problem, you can save the font with the document by enabling the Embed Fonts Using TrueDoc in the Save Drawing dialog box. Click File, Save As, and enable the Embed Fonts Using TrueDoc check box.

{button ,AL(`CLS_StructFontProperties')} **Related Topics**

StructFontProperties.Name

Property **Name** AS String

StructFontProperties

Description

The **Name** property returns or sets a string value that uniquely identifies a font in CorelDRAW. You can create and edit custom characters. The unique design of a character set is called its typeface. A font is a complete set of characters that share a common typeface -uppercase and lowercase letters, numbers, punctuation marks, and symbols.

A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font.

Example

The following code example displays the name of the active shape's text font in a message box:

```
Sub FontName()
With ActiveShape.Text.FontProperties
    MsgBox .Name
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Name')} **Related Topics**

cdrNormalFontStyle=0

cdrBoldFontStyle=1

cdrItalicFontStyle=2

cdrBoldItalicFontStyle=3

cdrThinFontStyle=4

cdrThinItalicFontStyle=5

cdrExtraLightFontStyle=6

cdrExtraLightItalicFontStyle=7

cdrMediumFontStyle=8

cdrMediumItalicFontStyle=9

cdrSemiBoldFontStyle=10

cdrSemiBoldItalicFontStyle=11

cdrExtraBoldFontStyle=12

cdrExtraBoldItalicFontStyle=13

cdrHeavyFontStyle=14

cdrHeavyItalicFontStyle=15

cdrMixedFontStyle=16

StructFontProperties.Style

Property **Style** AS **cdrFontStyle**

StructFontProperties

Description

The **Style** property returns or sets a value that identifies the style of a font in CorelDRAW. A style is a set of attributes that controls the appearance of a specific type of object. There are three style types: Graphic styles, Artistic text styles, and Paragraph text styles. You can use the styles in any CorelDRAW templates, or create and save your own styles.

The unique design of a character set is called its typeface. A font is a complete set of characters that share a common typeface -uppercase and lowercase letters, numbers, punctuation marks, and symbols. A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font.

The **Style** property returns a value of cdrFontStyle.

Example

The following code example sets the font style of the active shape's text to bold and italic. The **Underline** property is set to return no line underneath the text and the **Strikethru** property is set to return a thin double line through the text:

```
Sub FontProp()
With ActiveShape.Text.FontProperties
    .Style = cdrBoldItalicFontStyle
    .Underline = cdrNoFontLine
    .Strikethru = cdrDoubleThinFontLIne
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Style')} **Related Topics**

StructFontProperties.Size

Property **Size** AS Single

StructFontProperties

Description

The **Size** property returns or sets a numerical value that indicates a font size in CorelDRAW. You can increase or decrease the size of text and the default unit of text measurement is points.

The unique design of a character set is called its typeface. A font is a complete set of characters that share a common typeface -uppercase and lowercase letters, numbers, punctuation marks, and symbols. A font is a single style, weight, and size of a typeface, such as Times Roman bold, 10 point. Times Roman 18 point is a different font.

You can also change the font type and size using the Property Bar.

Example

The following code example sets the size of the text in the active shape to 46 point text. The position of the text is set to normal with the **Position** property, and the **RangeKerning** property sets the range value at 200, which adjusts the white space between text characters. The **Overscore** property is set to return no line above the text in the active shape:

```
Sub TextProp()
With ActiveShape.Text.FontProperties
    .Size = 46
    .Position = cdrNormalFontPosition
    .RangeKerning = 200
    .Overscore = cdrNoFontLine
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Size')} **Related Topics**

StructFontProperties.Underline

Property **Underline** AS **cdrFontLine**

StructFontProperties

Description

The **Underline** property returns or sets the type of line that appears under text in CorelDRAW. You can change the thickness of underlines, overscores, and strikethroughs. You can also change the distance between text and a line. By changing the line properties of underlines, overscores, and strikethroughs, you can further emphasize text.

You can also underline text by clicking the **Underline** button on the Property Bar, which lets you add an underline to the selected text or remove it from text. Text is underlined when the button appears pressed.

The **Underline** property returns a value of cdrFontLine.

Example

The following code example sets the **Style** property of the active shape's text to bold and italic. The **Underline** property is set to return no line underneath the text and the **Strikethru** property is set to return a thin double line through the text:

```
Sub FontProp()
With ActiveShape.Text.FontProperties
    .Style = cdrBoldItalicFontStyle
    .Underline = cdrNoFontLine
    .Strikethru = cdrDoubleThinFontLIne
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Underline')} **Related Topics**

StructFontProperties.Overscore

Property **Overscore** AS **cdrFontLine**

StructFontProperties

Description

The **Overscore** property returns or sets the type of line that appears above text in CorelDRAW. You can change the thickness of underlines, overscores, and strikethroughs. You can also change the distance between text and a line. By changing the line properties of underlines, overscores, and strikethroughs, you can further emphasize text.

The **Overscore** property returns a value of cdrFontLine.

Example

The following code example sets the size of the text in the active shape to 46 point text with the **Size** property. The position of the text is set to normal with the **Position** property, and the **RangeKerning** property sets the range value at 200, which adjusts the white space between text characters. The **Overscore** property is set to return no line above the text in the active shape:

```
Sub TextProp()
With ActiveShape.Text.FontProperties
    .Size = 46
    .Position = cdrNormalFontPosition
    .RangeKerning = 200
    .Overscore = cdrNoFontLine
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Overscore')} **Related Topics**

StructFontProperties.Strikethru

Property **Strikethru** AS **cdrFontLine**

StructFontProperties

Description

The **Strikethrough** property returns or sets the type of line that appears through text in CorelDRAW. You can change the thickness of underlines, overscores, and strikethroughs. You can also change the distance between text and a line. By changing the line properties of underlines, overscores, and strikethroughs, you can further emphasize text.

The **Strikethrough** property returns a value of cdrFontLine.

Example

The following code example sets the **Style** property of the active shape's text to bold and italic. The **Underline** property is set to return no line underneath the text and the **Strikethru** property is set to return a thin double line through the text:

```
Sub FontProp()
With ActiveShape.Text.FontProperties
    .Style = cdrBoldItalicFontStyle
    .Underline = cdrNoFontLine
    .Strikethru = cdrDoubleThinFontLIne
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Strikethru')} **Related Topics**

StructFontProperties.Uppercase

Property **Uppercase** AS **cdrFontCase**

StructFontProperties

Description

The **Uppercase** property returns or sets the case of text in CorelDRAW. For example, text can be uppercase, lowercase, or mixed case. The **Uppercase** property capitalizes all letters in a text selection in CorelDRAW.

The **Uppercase** property returns a value of cdrFontCase.

Example

The following code example capitalizes all text characters in the active shape using the **Uppercase** property:

```
Sub TextCaps()
With ActiveShape.Text.FontProperties
    .Uppercase = cdrAllCapsFontCase
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Uppercase')} **Related Topics**

cdrNormalFontPosition=0

cdrSubscriptFontPosition=1

cdrSuperscriptFontPosition=2

cdrMixedFontPosition=3

StructFontProperties.Position

Property **Position** AS **cdrFontPosition**

StructFontProperties

Description

The **Position** property returns or sets the position of text in CorelDRAW. Text can appear in several positions in relation to the text baseline, he invisible, horizontal line on which all the letters of a line sit. For example, text can be normal (on the baseline), subscript (below the baseline), superscript (above the baseline), or a combination of all three types.

The **Position** property returns a value of cdrFontPosition.

Example

The following code example sets the size of the text in the active shape to 46 point text with the **Size** property. The position of the text is set to normal with the **Position** property, and the **RangeKerning** property sets the range value at 200, which adjusts the white space between text characters. The **Overscore** property is set to return no line above the text in the active shape:

```
Sub TextProp()
With ActiveShape.Text.FontProperties
    .Size = 46
    .Position = cdrNormalFontPosition
    .RangeKerning = 200
    .Overscore = cdrNoFontLine
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_Position')} **Related Topics**

StructFontProperties.RangeKerning

Property **RangeKerning** AS Long

StructFontProperties

Description

The **RangeKerning** property returns or sets a numerical value that indicates the amount of white space between text chacacters in CorelDRAW. Kerning balances the optical space with other letters in a word or line. Kerning differs from spacing in that it affects only the white space between the specified characters.

This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 1000; the minimum percentage value is -100.

Example

The following code example sets the size of the text in the active shape to 46 point text with the **Size** property. The position of the text is set to normal with the **Position** property, and the **RangeKerning** property sets the range value at 200, which adjusts the white space between text characters. The **Overscore** property is set to return no line above the text in the active shape:

```
Sub TextProp()
With ActiveShape.Text.FontProperties
    .Size = 46
    .Position = cdrNormalFontPosition
    .RangeKerning = 200
    .Overscore = cdrNoFontLine
End With
End Sub
```

{button ,AL(`CLS_StructFontProperties;FNC_RangeKerning')} **Related Topics**

StructFontProperties.Fill

Property **Fill** AS **Fill**

StructFontProperties

Description

The **Fill** property returns or sets the fill value of text in CorelDRAW. Fills are colors, bitmaps, color gradients, or patterns that are applied to areas of your image.

In CorelDRAW, fills can be applied to any drawn object or curve. In Corel PHOTO-PAINT, fills can be applied to the contents of rectangles, polygons, etc., but are more often applied to portions of your bitmap image using the Fill tool.

{button ,AL(`CLS_StructFontProperties;FNC_Fill')} **Related Topics**

StructFontProperties.Outline

Property **Outline** AS **Outline**

StructFontProperties

Description

The **Outline** property returns or sets the outline of text in CorelDRAW. An outline is the line that defines the shape of an object. You can change outline attributes, including color, width, size, and shape, using the options in the Outline Tool button in CorelDRAW.

Example

Example of usage goes here

{button ,AL(`CLS_StructFontProperties;FNC_Outline')} **Related Topics**

StructAlignProperties properties

StructAlignProperties Legend

- Alignment
- CharacterRotation
- FirstLineIndent
- HorizontalCharacterShift
- LeftIndent
- MaxCharacterSpacing
- MaxWordSpacing
- MinWordSpacing
- RightIndent
- VerticalCharacterShift

StructAlignProperties

Class **StructAlignProperties**

Properties Referenced by

The **StructAlignProperties** class defines the alignment characteristics of text objects in a text frame and describes the look and behavior of the alignment through its properties and methods. Alignment changes the position of text within a selected text frame. For example, text can be aligned to the left, right or center of a text frame.

In CorelDRAW, you create Paragraph text with the Text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. Paragraph formatting features let you flow text between frames and columns, create bulleted lists, set tabs and indents, and add drop caps. CorelDRAW automatically applies the default Paragraph text style, which you can change using the Styles Manager.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

{button ,AL(`CLS_StructAlignProperties')} **Related Topics**

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

In CorelDRAW, you create Paragraph text with the Text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. Paragraph formatting features let you flow text between frames and columns, create bulleted lists, set tabs and indents, and add drop caps. CorelDRAW automatically applies the default Paragraph text style, which you can change using the Styles Manager.

StructAlignProperties.Alignment

Property **Alignment** AS **cdrAlignment**

StructAlignProperties

Description

The **Alignment** property sets or returns the alignment of paragraph text in a text frame in CorelDRAW. Alignment changes the position of text within a selected text frame. For example, text can be aligned to the left, right or center of a text frame.

The **Alignment** property returns a value of cdrAlignment.

Example

The following code example aligns all text in the active shape to the left of the text frame, sets the indent value of the first line of text, in document units and sets the left indent value of the entire text frame, in document units. It is important to note that the unit of measurement in each document may vary:

```
Sub TextAlign()
With ActiveShape.Text.AlignProperties
    .Alignment = cdrLeftAlignment
    .FirstLineIndent = 2
    .LeftIndent = 1
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_Alignment')} **Related Topics**

StructAlignProperties.FirstLineIndent

Property **FirstLineIndent** AS Double

StructAlignProperties

Description

The **FirstLineIndent** property returns or sets a numeric value that indents the first line of paragraph text in a text frame in CorelDRAW. You can change the space between a Paragraph text frame and its text by indenting. You can indent an entire paragraph, the first line of a paragraph, all but the first line of a paragraph (a hanging indent), or indent from the left or right side of the frame. You can also remove an indent without deleting or retyping.

The **FirstLineIndent** property only aligns only the first line of text in a paragraph text frame and it is measured in document units

Example

The following code example aligns all text in the active shape to the left of the text frame, sets the indent value of the first line of text, in document units and sets the left indent value of the entire text frame, in document units. It is important to note that the unit of measurement in each document may vary:

```
Sub TextAlign()
With ActiveShape.Text.AlignProperties
    .Alignment = cdrLeftAlignment
    .FirstLineIndent = 2
    .LeftIndent = 1
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_FirstLineIndent')} **Related Topics**

StructAlignProperties.LeftIndent

Property **LeftIndent** AS Double

StructAlignProperties

Description

The **LeftIndent** property returns or sets a numeric value that indents the left side of paragraph text in a text frame in CorelDRAW. You can change the space between a Paragraph text frame and its text by indenting. You can indent an entire paragraph, the first line of a paragraph, all but the first line of a paragraph (a hanging indent), or indent from the left or right side of the frame. You can also remove an indent without deleting or retyping.

The **LeftIndent** property only aligns only the left side of text in a paragraph text frame and it is measured in document units

Example

The following code example aligns all text in the active shape to the left of the text frame, sets the indent value of the first line of text, in document units and sets the left indent value of the entire text frame, in document units. It is important to note that the unit of measurement in each document may vary:

```
Sub TextAlign()
With ActiveShape.Text.AlignProperties
    .Alignment = cdrLeftAlignment
    .FirstLineIndent = 2
    .LeftIndent = 1
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_LeftIndent')} **Related Topics**

StructAlignProperties.RightIndent

Property **RightIndent** AS Double

StructAlignProperties

Description

The **RightIndent** property returns or sets a numeric value that indents the right side of paragraph text in a text frame in CorelDRAW. You can change the space between a Paragraph text frame and its text by indenting. You can indent an entire paragraph, the first line of a paragraph, all but the first line of a paragraph (a hanging indent), or indent from the left or right side of the frame. You can also remove an indent without deleting or retyping.

The **RightIndent** property only aligns only the right side of text in a paragraph text frame and it is measured in document units

Example

The following code example aligns all text in the active shape to the left of the text frame, sets the indent value of the first line of text, in document units and sets the right indent value of the entire text frame, in document units. It is important to note that the unit of measurement in each document may vary:

```
Sub TextAlign()
With ActiveShape.Text.AlignProperties
    .Alignment = cdrLeftAlignment
    .FirstLineIndent = 2
    .RightIndent = 1
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_RightIndent')} **Related Topics**

StructAlignProperties.MaxWordSpacing

Property **MaxWordSpacing** AS Single

StructAlignProperties

Description

The **MaxWordSpacing** property returns or sets a numerical value of the space that exists between words in the paragraph text of a text frame in CorelDRAW. This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is 0.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **MaxWordSpacing** sets or returns the maximum amount of space that can exist between words in a text frame.

Example

The following code example sets the maximum and minimum word spacing for the paragraph text in the active shape, as well as the maximum character spacing value for text in the active shape:

```
Sub TextMax()
With ActiveShape.Text.AlignProperties
    .MaxCharacterSpacing = 500
    .MinWordSpacing = 100
    .MaxWordSpacing = 500
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_MaxWordSpacing')} **Related Topics**

StructAlignProperties.MinWordSpacing

Property **MinWordSpacing** AS Single

StructAlignProperties

Description

The **MinWordSpacing** property returns or sets a numerical value of the space that exists between words in the paragraph text of a text frame in CorelDRAW. This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is 0.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **MinWordSpacing** sets or returns the minimum amount of space that can exist between words in a text frame.

Example

The following code example sets the maximum and minimum word spacing for the paragraph text in the active shape, as well as the maximum character spacing value for text in the active shape:

```
Sub TextMax()
With ActiveShape.Text.AlignProperties
    .MaxCharacterSpacing = 500
    .MinWordSpacing = 100
    .MaxWordSpacing = 500
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_MinWordSpacing')} **Related Topics**

StructAlignProperties.MaxCharacterSpacing

Property **MaxCharacterSpacing** AS Single

StructAlignProperties

Description

The **MaxCharacterSpacing** property returns or sets a numerical value of the space that exists between characters in the paragraph text of a text frame in CorelDRAW. .This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is -100.

A text character is a single text component. For example, in the text string "CorelDRAW Help", the character count is 14. Even though there are only 13 text characters, the space between "CorelDRAW" and "Help" is counted as a character.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **MaxCharacterSpacing** sets or returns the maximum amount of space that can exist between characters in a text frame.

Example

The following code example sets the maximum and minimum word spacing for the paragraph text in the active shape, as well as the maximum character spacing value for text in the active shape:

```
Sub TextMax()
With ActiveShape.Text.AlignProperties
    .MaxCharacterSpacing = 500
    .MinWordSpacing = 100
    .MaxWordSpacing = 500
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_MaxCharacterSpacing')} **Related Topics**

StructAlignProperties.HorizontalCharacterShift

Property **HorizontalCharacterShift** AS Long

StructAlignProperties

Description

The **HorizontalCharacterShift** property returns or sets a numerical value that horizontally moves a selected character in a text frame in CorelDRAW. Shifting individual and multiple Artistic text and Paragraph text objects vertically and horizontally can add an interesting effect to text. You can also rotate characters. When characters are shifted in relation to the baseline, which is the invisible, horizontal line on which all the letters of a line sit.

You can return characters that have been shifted horizontally to the baseline and maintain a vertical shift or rotation. You can also remove vertical shifts and rotations. You can also shift characters with the Shape tool by typing values in the Horizontal Shift box and Vertical Shift box on the Property Bar.

The value of a horizontal character shift is measured as a percentage of the character's point size. For example, if `.HorizontalCharacterShift=50`, the character is horizontally shifted a distance that is 50% of the character's point size.

Example

The following code example rotates all characters in the active shape 90, and shifts all characters in the active shape vertically and horizontally, 50% of their point sizes:

```
Sub TextShift()
With ActiveShape.Text.AlignProperties
    .CharacterRotation = 90
    .HorizontalCharacterShift = 50
    .VerticalCharacterShift = 50
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_HorizontalCharacterShift')} **Related Topics**

A point is a unit of measurement used primarily in typesetting to design type sizes. There are approximately 72 points (pts) to an inch and 12 points to a pica.

StructAlignProperties.VerticalCharacterShift

Property **VerticalCharacterShift** AS Long

StructAlignProperties

Description

The **VerticalCharacterShift** property returns or sets a numerical value that vertically moves a selected character in a text frame in CorelDRAW. Shifting individual and multiple Artistic text and Paragraph text objects vertically and horizontally can add an interesting effect to text. You can also rotate characters. When characters are shifted in relation to the baseline, which is the invisible, horizontal line on which all the letters of a line sit.

You can return characters that have been shifted horizontally to the baseline and maintain a vertical shift or rotation. You can also remove vertical shifts and rotations. You can also shift characters with the Shape tool by typing values in the Horizontal Shift box and Vertical Shift box on the Property Bar.

The value of a vertical character shift is measured as a percentage of the character's point size. For example, if `.VerticalCharacterShift=50`, the character is vertically shifted a distance that is 50% of the character's point size.

Example

The following code example rotates all characters in the active shape 90, and shifts all characters in the active shape vertically and horizontally, 50% of their point sizes:

```
Sub TextShift()
With ActiveShape.Text.AlignProperties
    .CharacterRotation = 90
    .HorizontalCharacterShift = 50
    .VerticalCharacterShift = 50
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_VerticalCharacterShift')} **Related Topics**

StructAlignProperties.CharacterRotation

Property **CharacterRotation** AS Single

StructAlignProperties

Description

The **CharacterRotation** property returns or sets a numerical value that rotates a character in the paragraph text of a text frame in CorelDRAW. Rotation is the repositioning or reorientation of an object by turning it around its center of rotation, the point around which an object rotates.

You can also rotate characters with the Shape tool by typing a value in the Angle of Rotation box on the Property Bar. The **CharacterRotation** property is measured in degrees. The maximum value is 360.

Example

The following code example rotates all characters in the active shape 90, and shifts all characters in the active shape vertically and horizontally, 50% of their point sizes:

```
Sub TextShift()
With ActiveShape.Text.AlignProperties
    .CharacterRotation = 90
    .HorizontalCharacterShift = 50
    .VerticalCharacterShift = 50
End With
End Sub
```

{button ,AL(`CLS_StructAlignProperties;FNC_CharacterRotation')} **Related Topics**

StructSpaceProperties properties

StructSpaceProperties Legend

- AfterParagraphSpacing
- BeforeParagraphSpacing
- CharacterSpacing
- LineSpacing
- LineSpacingType
- WordSpacing

StructSpaceProperties

Class **StructSpaceProperties**

Properties Referenced by

The **StructSpaceProperties** class defines the spacing characteristics of text objects in a text frame and describes the look and behavior of the spacing through its properties and methods. Spacing sets the distance between characters, words, paragraphs, and lines in a text frame.

In CorelDRAW, you create Paragraph text with the Text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. Paragraph formatting features let you flow text between frames and columns, create bulleted lists, set tabs and indents, and add drop caps. CorelDRAW automatically applies the default Paragraph text style, which you can change using the Styles Manager.

A paragraph text frame is the rectangle that contains a block of paragraph text created using the text tool. Paragraph text is used to add large blocks of text for ads, brochures, and other text-intensive projects. CorelDRAW automatically applies the default Paragraph text style.. A text frame may exist alone, or may be linked to other text frames. Linking paragraph text frames directs the flow of text from one frame to the other, if the amount of text exceeds the original frame. When you shrink a frame, enlarge a frame, or change text size, the amount of text in the next frame adjusts automatically. You can remove links or change the direction of flow.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines. When you space characters manually, you can change default settings to display or hide character outlines. You can also align Artistic text horizontally.

You can change the space between all characters, all words, or characters and words proportionately. You can also kern specific characters. Kerning balances the optical space with other letters in a word or line. Kerning differs from spacing in that it affects only the white space between the specified characters.

{button ,AL(`CLS_StructSpaceProperties')} **Related Topics**

StructSpaceProperties.CharacterSpacing

Property **CharacterSpacing** AS Single

StructSpaceProperties

Description

The **CharacterSpacing** property returns or sets a numerical value of the space that exists between characters in the paragraph text of a text frame in CorelDRAW. This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is -100.

A text character is a single text component. For example, in the text string "CorelDRAW Help", the character count is 14. Even though there are only 13 text characters, the space between "CorelDRAW" and "Help" is counted as a character.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **CharacterSpacing** property sets or returns the amount of space that can exist between characters in a text frame. The value set with the **CharacterSpacing** property cannot exceed the maximum character spacing limit, set with the MaxCharacterSpacing property.

Example

The following code example sets the character and word spacing values for the paragraph text in the active shape of CorelDRAW. These numerical spacing values represent percentages of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is -100:

```
Sub TextSpacing()
With ActiveShape.Text.SpaceProperties
    .CharacterSpacing = 150
    .WordSpacing = 250
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_CharacterSpacing')} **Related Topics**

StructSpaceProperties.WordSpacing

Property **WordSpacing** AS Single

StructSpaceProperties

Description

The **WordSpacing** property returns or sets a numerical value of the space that exists between words in the paragraph text of a text frame in CorelDRAW. .This value represents a percentage of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is 0.

By changing the spacing of Artistic text and Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **WordSpacing** property sets or returns the amount of space that can exist between words in a text frame. The value set with the **WordSpacing** property cannot exceed the maximum or minimum word spacing limits, set with the MaxWordSpacing and MinWordSpacing properties.

Example

The following code example sets the character and word spacing values for the paragraph text in the active shape of CorelDRAW. These numerical spacing values represent percentages of the space character (the space inserted when you press SPACEBAR). The maximum percentage value is 2000; the minimum percentage value is -100:

```
Sub TextSpacing()
With ActiveShape.Text.SpaceProperties
    .CharacterSpacing = 150
    .WordSpacing = 250
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_WordSpacing')} **Related Topics**

StructSpaceProperties.LineSpacing

Property **LineSpacing** AS Single

StructSpaceProperties

Description

The **LineSpacing** property returns or sets a numerical value of line spacing in the paragraph text of a text frame in CorelDRAW. A line can be a continuation of wrapped text from a previous line or it may be a new line created by a carriage return.

The **LineSpacing** value can vary in range depending on the type of line spacing selected with the LineSpacingType property. For example, line spacing can be measured as a percentage of character height, points, or a percentage of point size.

By changing the spacing of lines in Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

For Paragraph text, the space applies to lines of text within the same paragraph. You can also change the spacing before and after paragraphs for Paragraph text.

Example

The following code example sets the line spacing type of paragraph text in the active shape to point line spacing, making the line spacing value a percentage of the character font size. The line spacing value is set to 100 points:

```
Sub TextLineSpacing()
With ActiveShape.Text.SpaceProperties
    .LineSpacingType = cdrPointLineSpacing
    .LineSpacing = 100
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_LineSpacing')} **Related Topics**

StructSpaceProperties.LineSpacingType

Property **LineSpacingType** AS **cdrLineSpacingType**

StructSpaceProperties

Description

The **LineSpacingType** property returns or sets the line spacing type in the paragraph text of a text frame in CorelDRAW. A line can be a continuation of wrapped text from a previous line or it may be a new line created by a carriage return.

The **LineSpacing** property value can vary in range depending on the type of line spacing selected with the **LineSpacingType** property. For example, line spacing can be measured as a percentage of character height, points, or a percentage of point size.

By changing the spacing of Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

For Paragraph text, the space applies to lines of text within the same paragraph. You can also change the spacing before and after paragraphs for Paragraph text.

The **LineSpacingType** property returns a value of cdrLineSpacingType.

Example

The following code example sets the line spacing type of paragraph text in the active shape to point line spacing, making the line spacing value a percentage of the character font size. The line spacing value is set to 100 points:

```
Sub TextLineSpacing()
With ActiveShape.Text.SpaceProperties
    .LineSpacingType = cdrPointLineSpacing
    .LineSpacing = 100
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_LineSpacingType')} **Related Topics**

cdrPercentOfCharacterHeightLineSpacing=0

cdrPointLineSpacing=1

cdrPercentOfPointSizeLineSpacing=2

cdrMixedLineSpacing=3

StructSpaceProperties.BeforeParagraphSpacing

Property **BeforeParagraphSpacing** AS Single

StructSpaceProperties

Description

The **BeforeParagraphSpacing** property returns or sets a numerical value that indicates the amount of space before paragraph text in a text frame in CorelDRAW. Unlike lines, which can contain wrapped text from the previous line, a paragraph is marked by a carriage return, which signals the end of one paragraph and the beginning of another paragraph.

By changing the spacing before Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **BeforeParagraphSpacing** value can vary in range depending on the type of paragraph spacing selected. For example, spacing can be measured as a percentage of character height, points, or a percentage of point size.

Example

The following code example sets the spacing before and after paragraph text in the active shape of CorelDRAW. The values for spacing before and after paragraph text in this example are point values, as defined by setting the **LineSpacingType** property:

```
Sub TextParaSpace()
With ActiveShape.Text.SpaceProperties
    .LineSpacingType = cdrPointLineSpacing
    .BeforeParagraphSpacing = 200
    .AfterParagraphSpacing = 200
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_BeforeParagraphSpacing')} **Related Topics**

StructSpaceProperties.AfterParagraphSpacing

Property **AfterParagraphSpacing** AS Single

StructSpaceProperties

Description

The **AfterParagraphSpacing** property returns or sets a numerical value that indicates the amount of space after paragraph text in a text frame in CorelDRAW. Unlike lines, which can contain wrapped text from the previous line, a paragraph is marked by a carriage return, which signals the end of one paragraph and the beginning of another paragraph.

By changing the spacing after Paragraph text, you can enhance the text and make it more readable. You can change the spacing between characters, words, and lines.

The **AfterParagraphSpacing** value can vary in range depending on the type of paragraph spacing selected. For example, spacing can be measured as a percentage of character height, points, or a percentage of point size.

Example

The following code example sets the spacing before and after paragraph text in the active shape of CorelDRAW. The values for spacing before and after paragraph text in this example are point values, as defined by setting the **LineSpacingType** property:

```
Sub TextParaSpace()
With ActiveShape.Text.SpaceProperties
    .LineSpacingType = cdrPointLineSpacing
    .BeforeParagraphSpacing = 200
    .AfterParagraphSpacing = 200
End With
End Sub
```

{button ,AL(`CLS_StructSpaceProperties;FNC_AfterParagraphSpacing')} **Related Topics**

StructHyphenationSettings properties

StructHyphenationSettings Legend

- BreakCapitalized
- HotZone
- MinCharactersAfter
- MinCharactersBefore
- MinWordLength
- UseAutomaticHyphenation

StructHyphenationSettings

Class **StructHyphenationSettings**

Properties Referenced by

The **StructHyphenationSettings** class defines the hyphen characteristics of text objects in a text frame and describes the look and behavior of the hyphenation through its properties and methods. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

Hyphenation can be useful when you work with columns or have limited space for text. By enabling hyphenation, CorelDRAW automatically divides words at the end of lines instead of wrapping them to the next line. You can apply hyphenation to a Paragraph text frame or to selected paragraphs within a frame.

By customizing the hyphenation settings, you can control when hyphenation occurs. You can customize hyphenation settings by applying hyphenation to words containing capital letters and specifying the minimum number of characters required in a word for hyphenation to occur. You can also specify the minimum number of letters before and after a hyphen, as well as the distance from the right margin that CorelDRAW starts hyphenating words (also called the hot zone).

{button ,AL(`CLS_StructHyphenationSettings')} **Related Topics**

StructHyphenationSettings.UseAutomaticHyphenation

Property **UseAutomaticHyphenation** AS Boolean

StructHyphenationSettings

Description

The **UseAutomaticHyphenation** property returns or sets a True or False value that indicates if the automatic hyphenation of text is enabled in CorelDRAW. If the **UseAutomaticHyphenation** property is True, CorelDRAW automatically hyphenates text. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

You can apply automatic hyphenation to all paragraph text in a document or to specific text within a paragraph text frame. If you set the default text style to automatically hyphenate text, the existing text in your document remains unaffected. Only subsequent frames you create will use automatic hyphenation. If you select specific text and set the automatic hyphenation property, the selected text applies the new setting.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_UseAutomaticHyphenation')} **Related Topics**

StructHyphenationSettings.BreakCapitalized

Property **BreakCapitalized** AS Boolean

StructHyphenationSettings

Description

The **BreakCapitalized** property returns or sets a True or False value that indicates if the automatic hyphenation of text containing capital letters is enabled in CorelDRAW. If the **BreakCapitalized** property is True, CorelDRAW automatically hyphenates words with initial or all capital letters.

This property sets capital letters apart from normal characters with a hyphen. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

In order to return or set the **BreakCapitalized** property, the **UseAutomaticHyphenation** property must be set to True.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_BreakCapitalized')} **Related Topics**

StructHyphenationSettings.HotZone

Property **HotZone** AS Double

StructHyphenationSettings

Description

The **HotZone** property returns or sets a value that defines the distance from the right margin of a paragraph text frame where CorelDRAW begins hyphenating text. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text.

The **HotZone** property is numerical value measured in inches. For example, if the **HotZone** property value is 2, CorelDRAW begins hyphenating text 2 inches from the right margin.

In order to return or set the **HotZone** property, the **UseAutomaticHyphenation** property must be set to True.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_HotZone')} **Related Topics**

StructHyphenationSettings.MinWordLength

Property **MinWordLength** AS Long

StructHyphenationSettings

Description

The **MinWordLength** property returns or sets a numerical value that indicates the minimum number of characters a word must contain in order to hyphenate paragraph text in CorelDRAW. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text. A text character is a single text component.

For example, if the **MinWordLength** is set to 5, a word in a paragraph text frame must contain at least 5 characters in order to apply hyphenation.

In order to return or set the **MinWordLength** property, the **UseAutomaticHyphenation** property must be set to True.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_MinWordLength')} **Related Topics**

StructHyphenationSettings.MinCharactersBefore

Property **MinCharactersBefore** AS Long

StructHyphenationSettings

Description

The **MinCharactersBefore** property returns or sets a numerical value that indicates the minimum number of characters that must exist before a hyphen in paragraph text in CorelDRAW. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text. A text character is a single text component.

For example, if the **MinCharactersBefore** is set to 3, a word in a paragraph text frame must contain at least 3 characters before a hyphen in order to apply hyphenation.

In order to return or set the **MinCharactersBefore** property, the **UseAutomaticHyphenation** property must be set to True.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_MinCharactersBefore')} **Related Topics**

StructHyphenationSettings.MinCharactersAfter

Property **MinCharactersAfter** AS Long

StructHyphenationSettings

Description

The **MinCharactersAfter** property returns or sets a numerical value that indicates the minimum number of characters that must exist after a hyphen in paragraph text in CorelDRAW. Text can use hyphens to join words or to indicate the continuation of a word from one line of text to another line of text. A text character is a single text component.

For example, if the **MinCharactersAfter** is set to 4, a word in a paragraph text frame must contain at least 4 characters before a hyphen in order to apply hyphenation.

In order to return or set the **MinCharactersAfter** property, the **UseAutomaticHyphenation** property must be set to True.

Example

The following code example sets enables automatic hyphenation in the active shape of CorelDRAW, and enables the **BreakCapitalized** property - words containing capital letters are set apart from normal text. Hyphenation only applies to words that contain at least 6 characters, and there must be at least 3 characters before and after a hyphen. The **HotZone** property is set to 1 inch, enabling CorelDRAW to begin hyphenating words 1 inch from the right margin:

```
Sub TextHyphen()
With ActiveShape.Text.HyphenationSettings
    .UseAutomaticHyphenation = True
    .BreakCapitalized = True
    .HotZone = 1
    .MinCharactersBefore = 3
    .MinCharactersAfter = 3
    .MinWordLength = 6
End With
End Sub
```

{button ,AL(`CLS_StructHyphenationSettings;FNC_MinCharactersAfter')} **Related Topics**

StructPaletteOptions properties

StructPaletteOptions Legend

- ColorSensitive
- DitherIntensity
- DitherType
- Importance
- Lightness
- NumColors
- PaletteType
- Smoothing
- TargetColor
- ToleranceA
- ToleranceB

StructPaletteOptions

Class **StructPaletteOptions**

Properties Referenced By

{button ,AL(`CLS_StructPaletteOptions')} **Related Topics**

StructPaletteOptions.ColorSensitive

Property **ColorSensitive** As Boolean

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_ColorSensitive')} **Related Topics**

StructPaletteOptions.DitherIntensity

Property **DitherIntensity** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_DitherIntensity')} **Related Topics**

StructPaletteOptions.DitherType

Property **DitherType** As **cdrDitherType**

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_DitherType')} **Related Topics**

StructPaletteOptions.Importance

Property **Importance** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_Importance')} **Related Topics**

StructPaletteOptions.Lightness

Property **Lightness** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_Lightness')} **Related Topics**

StructPaletteOptions.NumColors

Property **NumColors** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_NumColors')} **Related Topics**

StructPaletteOptions.PaletteType

Property **PaletteType** As **cdrImagePaletteType**

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_PaletteType')} **Related Topics**

StructPaletteOptions.Smoothing

Property **Smoothing** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_Smoothing')} **Related Topics**

StructPaletteOptions.TargetColor

Property **TargetColor** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_TargetColor')} **Related Topics**

StructPaletteOptions.ToleranceA

Property **ToleranceA** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_ToleranceA')} **Related Topics**

StructPaletteOptions.ToleranceB

Property **ToleranceB** As Long

Member of **StructPaletteOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructPaletteOptions;FNC_ToleranceB')} **Related Topics**

TrapLayer properties

TrapLayer Legend

- Color
  - Density
  - Order
  - TrapType
- Type

TrapLayer

Class **TrapLayer**

Properties Referenced By

TrapLayer Class

{button ,AL(`CLS_TrapLayer')} **Related Topics**

TrapLayer.Color

Property **Color** As String

property Color

Member of **TrapLayer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayer;FNC_Color')} **Related Topics**

TrapLayer.Density

Property **Density** As Double

property Density

Member of **TrapLayer**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayer;FNC_Density')} **Related Topics**

TrapLayer.Order

Property **Order** As Long

property Order

Member of **TrapLayer**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayer;FNC_Order')} **Related Topics**

TrapLayer.TrapType

Property **TrapType** As **PrnTrapType**

property TrapType

Member of **TrapLayer**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayer;FNC_TrapType')} **Related Topics**

TrapLayer.Type

Property **Type** As **PrnPlateType**

property Type

Member of **TrapLayer**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayer;FNC_Type')} **Related Topics**

TrapLayers properties

TrapLayers Legend

- Count

- Item

TrapLayers

Class **TrapLayers**

Properties Referenced By

TrapLayers Class

{button ,AL(`CLS_TrapLayers')} **Related Topics**

TrapLayers.Count

Property **Count** As Long

property Count

Member of **TrapLayers**

Read-Only

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayers;FNC_Count')} **Related Topics**

TrapLayers.Item

Property **Item**(ByVal **Index** As Long) As **TrapLayer**

property Item

Member of **TrapLayers**

Read-Only

Parameters	Description
Index	Description of Index goes here (**in**)

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_TrapLayers;FNC_Item')} **Related Topics**

URL properties

URL Legend

- Address
- AltComment
- BookMark
- Region
- TargetFrame

URL

Class **URL**

Properties Referenced By

URL Class

{button ,AL(`CLS_URL')} **Related Topics**

URL.Address

Property **Address** As String

Gets or sets URL Address

Member of **URL**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_URL;FNC_Address')} **Related Topics**

URL.AltComment

Property **AltComment** As String

Gets or sets URL ALT Comment

Member of **URL**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_URL;FNC_AltComment')} **Related Topics**

URL.BookMark

Property **BookMark** As String

Gets or sets URL Bookmark

Member of **URL**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_URL;FNC_BookMark')} **Related Topics**

URL.Region

Property **Region** As **cdrURLRegion**

Gets or sets URL Region

Member of **URL**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_URL;FNC_Region')} **Related Topics**

URL.TargetFrame

Property **TargetFrame** As String

Gets or sets URL Target Frame Name

Member of **URL**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_URL;FNC_TargetFrame')} **Related Topics**

StructExportOptions properties

StructExportOptions Legend

- AntiAliasingType
- Compression
- Dithered
- ImageType
- Overwrite
- ResolutionX
- ResolutionY
- SizeX
- SizeY
- Transparent
- UseColorProfile

StructExportOptions.Compression

Property **Compression** As **cdrCompressionType**

Member of **StructExportOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructExportOptions;FNC_Compression')} **Related Topics**

StructExportOptions

Class **StructExportOptions**

Properties Referenced by

The **StructExportOptions** class defines features of exporting objects in CorelDRAW and describes the look and behavior of the export process through its properties and methods.

If you want to save a file in a nonnative format, you must use the Export or Save As command to convert it to that file format. Both the Export and Save As Drawing dialog boxes let you choose the drive and folder where you wish to store your file. In CorelDRAW, use the Export command to access the bitmap filters, the RTF format, and word-processing file formats, in addition to the vector filters. The Export dialog box lets you choose to export files as compressed or uncompressed for file formats that support this feature.

The Export dialog box contains options that let you suppress filter dialog and choose to export files as compressed or uncompressed. It also lets you use the Export command to access the bitmap filters, the RTF format, and word-processing file formats, in addition to the vector filters. You can also embed an ICC profile, and control no white space, special characters in your filename.

Choosing Compression type lets you choose to export files as compressed or uncompressed for file formats that support this feature.

{button ,AL(`CLS_StructExportOptions')} **Related Topics**

StructExportOptions.Dithered

Property **Dithered** As Boolean

Member of **StructExportOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructExportOptions;FNC_Dithered')} **Related Topics**

StructExportOptions.Transparent

Property **Transparent** As Boolean

Member of **StructExportOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructExportOptions;FNC_Transparent')} **Related Topics**

StructExportOptions.UseColorProfile

Property **UseColorProfile** As Boolean

Member of **StructExportOptions**

Example

Example of usage goes here

```
Code line
```

{button ,AL(`CLS_StructExportOptions;FNC_UseColorProfile')} **Related Topics**

StructExportOptions.SizeX

Property **SizeX** AS Long

StructExportOptions

Description

The **SizeY** property returns or sets a numerical value that indicates the vertical proportions of an object that is exported in CorelDRAW. The **SizeX** and **SizeY** properties set the area of an object that is exported in CorelDRAW.

The **SizeY** property is measured in document units.

{button ,AL(`CLS_StructExportOptions;FNC_SizeX')} **Related Topics**

StructExportOptions.SizeY

Property **SizeY** AS Long

StructExportOptions

Description

The **SizeY** property returns or sets a numerical value that indicates the vertical proportions of an object that is exported in CorelDRAW. The **SizeX** and **SizeY** properties set the area of an object that is exported in CorelDRAW.

The **SizeY** property is measured in document units.

{button ,AL(`CLS_StructExportOptions;FNC_SizeY')} **Related Topics**

StructExportOptions.ResolutionX

Property **ResolutionX** AS Long

StructExportOptions

Description

The **ResolutionX** property returns or sets a numerical value that indicates the horizontal resolution level of an image that is exported in CorelDRAW. Resolution is the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size.

Image resolution refers to the spacing of pixels in the image and is measured in pixels per inch (ppi) or dots per inch (dpi).

When creating a graphic for Internet use, setting a horizontal and vertical resolution for your graphic ensures that it looks the same regardless of what application is used to display it.

{button ,AL(`CLS_StructExportOptions;FNC_ResolutionX')} **Related Topics**

StructExportOptions.ResolutionY

Property **ResolutionY** AS Long

StructExportOptions

Description

The **ResolutionY** property returns or sets a numerical value that indicates the vertical resolution level of an image that is exported in CorelDRAW. Resolution is the amount of detail and information that an image file contains, as well as the level of detail that an input, output or display device is capable of producing. When you work with bitmaps, resolution affects the quality of your final output and the file size.

Image resolution refers to the spacing of pixels in the image and is measured in pixels per inch (ppi) or dots per inch (dpi).

When creating a graphic for Internet use, setting a horizontal and vertical resolution for your graphic ensures that it looks the same regardless of what application is used to display it.

{button ,AL(`CLS_StructExportOptions;FNC_ResolutionY')} **Related Topics**

cdrNoAntiAliasing=0

cdrNormalAntiAliasing=1

cdrSupersampling=2

StructExportOptions.AntiAliasingType

Property **AntiAliasingType** AS **cdrAntiAliasingType**

StructExportOptions

Description

The **AntiAliasingType** property returns or sets a value that indicates the appearance of a bitmap image's edges when it is exported in CorelDRAW. Anti-aliasing is a method of smoothing the curved and diagonal edges contained in bitmap images. Anti-aliasing partially fills intermediate pixels along those edges to smooth the transition between the edge and the surrounding image. Anti-aliasing reduces or eliminates jagged edges.

The **AntiAliasingType** property returns a value of cdrAntiAliasingType.

{button ,AL(`CLS_StructExportOptions;FNC_AntiAliasingType')} **Related Topics**

StructExportOptions.Overwrite

Property **Overwrite** AS Boolean

StructExportOptions

Description

The **Overwrite** property returns or sets a True or False value that indicates in the object you are exporting will overwrite, or replace, an existing object with the same name.

If the **Overwrite** property is True, the most recently saved object replaces the object of the same name.

{button ,AL(`CLS_StructExportOptions;FNC_Overwrite')} **Related Topics**

cdrBlackAndWhiteImage=0

cdr16ColorsImage=1

cdrGrayscaleImage=2

cdrPalettedImage=3

cdrRGBColorImage=4

cdrCMYKColorImage=5

cdrDuotoneImage=6

StructExportOptions.ImageType

Property **ImageType** AS **cdrImageType**

StructExportOptions

Description

The **ImageType** property returns or sets the type of image that is exported in CorelDRAW. An image can be black and white, 16 colors, grayscale, paletted, RGB, CMYK or duotone.

The **ImageType** property returns a value of cdrImageType.

{button ,AL(`CLS_StructExportOptions;FNC_ImageType')} **Related Topics**

Duotone is an 8-bit color mode that displays images using 256 shades of up to four tones. An image in the Duotone color mode is simply a grayscale image that has been enhanced with one to four additional colors. Use the Duotone color mode to add a touch of color to grayscale images or to create interesting effects using tone curve settings. A duotone image can be monotone, duotone, tritone, or quadtone.

Paletted is an 8-bit color mode that displays images using up to 256 colors. You can convert a complex image to the paletted color mode to reduce file size and to more precisely control the colors used throughout the conversion process.

StructSaveAsOptions properties

StructSaveAsOptions Legend

- EmbedICCProfile
- EmbedVBAProject
- Filter
- IncludeCMXData
- Overwrite
- Range
- ThumbnailSize
- Version

StructSaveAsOptions

Class **StructSaveAsOptions**

Properties Referenced by

The **StructSaveAsOptions** class defines features of saving objects in CorelDRAW and describes the look and behavior of the save process through its properties and methods.

CorelDRAW lets you save your files in the way that best suits your file management needs. You can save only the selected objects of a drawing, make a copy of a drawing by saving it with a different name, or save a drawing in a format that is compatible with an earlier version of CorelDRAW. There are also several vector formats you can choose from.

CorelDRAW offers advanced options that let you assign notes, keywords, and thumbnails so you can find your files more easily in future sessions. You can save fonts, textures, blends, and extrudes with the file, or save references to them. These references reduce the size of your file, making it faster to open and save.

Before you close an active file, CorelDRAW asks whether or not you wish to save it. You can also close specific viewing windows, and close all open files or views using a single command.

{button ,AL(`CLS_StructSaveAsOptions')} **Related Topics**

StructSaveAsOptions.Filter

Property **Filter** AS **cdrFilter**

StructSaveAsOptions

Description

The **Filter** property returns or sets the filter that is used when saving objects in CorelDRAW. A filter is the name for an application that translates digital information from one form to another. Import/Export filters convert files from one format to another. When you select a file format in the Export dialog box of CorelDRAW, you automatically activate the appropriate filter application to perform the translation.

The **Filter** property returns a value of **cdrFilter**.

{button ,AL(`CLS_StructSaveAsOptions;FNC_Filter')} **Related Topics**

cdrCurrentVersion=0

cdrVersion5=5

cdrVersion6=6

cdrVersion7=7

cdrVersion8=8

cdrVersion9=9

cdrVersion10=10

StructSaveAsOptions.Version

Property **Version** AS **cdrFileVersion**

StructSaveAsOptions

Description

The **Version** property returns or sets the application version of CorelDRAW when saving an object. A version is an edition of CorelDRAW - it is a particular form of the application, differing from previous editions.

The **Version** property returns a value of cdrFileVersion.

{button ,AL(`CLS_StructSaveAsOptions;FNC_Version')} **Related Topics**

cdrNoThumbnail=0

cdr1KMonoThumbnail=1

cdr5KColorThumbnail=2

cdr10KColorThumbnail=3

StructSaveAsOptions.ThumbnailSize

Property **ThumbnailSize** AS **cdrThumbnailSize**

StructSaveAsOptions

Description

The **ThumbnailSize** property returns or sets the type of thumbnail version, if any, of an object in CorelDRAW. A thumbnail is a miniature, low-resolution version of an image or illustration. A thumbnail is also called a header.

Including a thumbnail when you save a file lets you see a representation of the drawing before you open it in a different application, such as Corel VENTURA. A thumbnail of the drawing is displayed in the Open dialog box of the other application and lets you verify that you have chosen the right file before you continue.

The **ThumbnailSize** property returns a value of cdrThumbnailSize.

{button ,AL(`CLS_StructSaveAsOptions;FNC_ThumbnailSize')} **Related Topics**

cdrNoThumbnail=0

cdr1KMonoThumbnail=1

cdr5KColorThumbnail=2

cdr10KColorThumbnail=3

cdrAllPages=0

cdrCurrentPage=1

cdrSelection=2

StructSaveAsOptions.Range

Property **Range** AS **cdrExportRange**

StructSaveAsOptions

Description

The **Range** property returns or sets the objects that are saved in CorelDRAW. You have the option of saving all or several pages in a document, a single page in a document, or a selection of objects within a document.

The **Range** property returns a value of cdrExportRange.

{button ,AL(`CLS_StructSaveAsOptions;FNC_Range')} **Related Topics**

StructSaveAsOptions.Overwrite

Property **Overwrite** AS Boolean

StructSaveAsOptions

Description

The **Overwrite** property returns or sets a True or False value that indicates in the object you are saving will overwrite, or replace, an existing object with the same name.

If the **Overwrite** property is True, the most recently saved object replaces the object of the same name.

{button ,AL(`CLS_StructSaveAsOptions;FNC_Overwrite')} **Related Topics**

StructSaveAsOptions.EmbedICCProfile

Property **EmbedICCProfile** AS Boolean

StructSaveAsOptions

Description

The **EmbedICCProfile** property returns or sets a True of False value that indicates if an ICC profile is embedded, or contained, within an object when it is saved with CorelDRAW. ICC is International Color Consortium, an organization that sets standards for device characterization. In color management, a device is a file that describes the color-producing characteristics of a device. Most color management software uses profiles that are in the ICC format.

If the **EmbedICCProfile** property is True, the ICC file is included when an object is saved with CorelDRAW.

{button ,AL(`CLS_StructSaveAsOptions;FNC_EmbedICCProfile')} **Related Topics**

StructSaveAsOptions.EmbedVBAProject

Property **EmbedVBAProject** AS Boolean

StructSaveAsOptions

Description

The **EmbedVBAProject** property returns or sets a True of False value that indicates if automated components of a Visual Basic for Applications (VBA) project are embedded, or contained, within an object when it is saved with CorelDRAW. VBA allows you to automate CorelDRAW tasks using the Visual Basic programming language. The integration of the VBA Editor lets you create projects that you can run in CorelDRAW. By incorporating VBA, Corel provides an international programming language to users.

If the **EmbedVBAProject** property is True, automation tasks and components are included when an object is saved with CorelDRAW.

{button ,AL(`CLS_StructSaveAsOptions;FNC_EmbedVBAProject')} **Related Topics**

StructSaveAsOptions.IncludeCMXData

Property **IncludeCMXData** AS Boolean

StructSaveAsOptions

Description

The **IncludeCMXData** property returns or sets a True of False value that indicates if CMX data is included when an object is saved with CorelDRAW. The CMX file format was originally developed to save files created in CorelDRAW with the data necessary to open and edit them in other Corel applications. Corel applications support version 5 6, 7 and 8 of the CMX file format.

If the **IncludeCMXData** property is True, the CMS data is included when an object is saved with CorelDRAW.

{button ,AL(`CLS_StructSaveAsOptions;FNC_IncludeCMXData')} **Related Topics**

AbsoluteIndex Property

Select one of the available subtopics below to see detailed help on **AbsoluteIndex** property

Object	Property description
Node	Gets the index of the node within the curve
Segment	Gets the index of the segment within the curve

Active Property

Select one of the available subtopics below to see detailed help on **Active** property

Object	Property description
AppWindow	Gets the CorelDRAW's window active status
Document	Gets if the page is active
Window	Gets if the window is active
Workspace	Gets the current workspace

ActiveLayer Property

Select one of the available subtopics below to see detailed help on **ActiveLayer** property

Object	Property description
Application	Gets the active layer
Document	Gets the reference to the active layer in the currently active document
Page	Gets the active layer on the page object

ActivePage Property

Select one of the available subtopics below to see detailed help on **ActivePage** property

Object	Property description
Application	Gets the reference to the currently active page
Document	Gets the reference to the active page in the currently active document

ActiveShape Property

Select one of the available subtopics below to see detailed help on **ActiveShape** property

Object	Property description
Application	Gets the active shape
Document	Gets the active shape

ActiveWindow Property

Select one of the available subtopics below to see detailed help on **ActiveWindow** property

Object	Property description
Application	Gets the reference to the currently active window
Document	Gets the reference to the active window in the currently active document

Amplitude Property

Select one of the available subtopics below to see detailed help on **Amplitude** property

Object	Property description
EffectPushPullDistortion	Gets or sets the amplitude of the PushPull distortion
EffectZipperDistortion	Gets or sets the amplitude of the Zipper distortion

Angle Property

Select one of the available subtopics below to see detailed help on **Angle** property

Object	Property description
EffectBlend	Gets or sets the blend angle
EffectTwisterDistortion	Gets or sets the distortion angle of the Twister distortion
FountainFill	Gets a fountain fill's angle
SeparationPlate	property Angle
StructFountainFillProperties	

Application Property

Select one of the available subtopics below to see detailed help on **Application** property

Object	Property description
ActiveView	Gets the application to which the object belongs
AddinHook	
AddIns	
Application	Gets the application to which the object belongs
AppWindow	Gets the application to which the object belongs
ArrowHeads	Gets the application to which the arrow head collection belongs
Clipboard	Gets the application to which the object belongs
CloneLink	Gets the application to which the object belongs
Color	Gets the application to which the object belongs
Colors	Gets the application to which the color collection belongs
CorelScriptFile	Gets the application to which the object belongs
DataField	Gets the application to which the object belongs
DataFields	Gets the application to which the object belongs
DataItem	Gets the application to which the object belongs
DataItems	Gets the application to which the object belongs
Document	Gets the application to which the object belongs
Documents	Gets the application to which the document collection belongs
Effect	Gets the application to which the object belongs
EffectArtistic	Gets the application to which the object belongs
EffectBlend	Gets the application to which the object belongs
EffectContour	Gets the application to which the object belongs
EffectControlPath	Gets the application to which the object belongs
EffectDistortion	Gets the application to which the object belongs
EffectDropShadow	Gets the application to which the object belongs
EffectEnvelope	Gets the application to which the object belongs
EffectExtrude	Gets the application to which the object belongs
EffectLens	Gets the application to which the object belongs
EffectPerspective	Gets the application to which the object belongs
EffectPushPullDistortion	Gets the application to which the object belongs
Effects	Gets the application to which the object belongs
EffectTextOnPath	Gets the application to which the object belongs
EffectTwisterDistortion	Gets the application to which the object belongs
EffectZipperDistortion	Gets the application to which the object belongs
ExtrudeVanishingPoint	Gets the application to which the object belongs
FontList	Gets the application to which the font list belongs
Grid	Gets the application to which the object belongs
Layer	Gets the application to which the object belongs
Layers	Gets the application to which the layer collection belongs

Node	Gets the application to which the node belongs
NodeRange	Gets the application to which the node range belongs
Nodes	Gets the application to which the node collection belongs
OutlineStyles	Gets the application to which the outlinestyle collection belongs
Page	Gets the application to which the object belongs
Pages	Gets the application to which the page collection belongs
Palette	Gets the application to which the object belongs
Palettes	Gets the application to which the palette collection belongs
Points	Gets the application to which the point collection belongs
PowerClip	Gets the application to which the object belongs
RecentFile	Gets the application to which the object belongs
RecentFiles	Gets the application to which the object belongs
Rulers	Gets the application to which the object belongs
Segment	Gets the application to which the segment belongs
SegmentRange	Gets the application to which the range belongs
Segments	Gets the application to which the segment collection belongs
Shape	Gets the application to which the object belongs
ShapePoint	Gets the application to which the point belongs
ShapeRange	Gets the application to which the shape range belongs
Shapes	Gets the application to which the shape collection belongs
SubPath	Gets the application to which the segment collection belongs
Subpaths	Gets the application to which the subpath collection belongs
Transparency	Gets the application to which the object belongs
View	Gets the application to which the object belongs
Views	Gets the application to which the object belongs
Window	Gets the application to which the object belongs
Windows	Gets the application to which the window collection belongs
Workspace	Gets the application to which the object belongs
Workspaces	Gets the application to which the workspace collection belongs

Bleed Property

Select one of the available subtopics below to see detailed help on **Bleed** property

Object	Property description
Page	Gets or sets the page's bleed
PDFVBASettings	

BlendType Property

Select one of the available subtopics below to see detailed help on **BlendType** property

Object	Property description
FountainFill	Gets or sets a fountain fill's blend type
StructFountainFillProperties	

Caption Property

Select one of the available subtopics below to see detailed help on **Caption** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window caption
CommandBarControl	Gets/sets the caption of the control
Window	Gets the window caption

CenterX Property

Select one of the available subtopics below to see detailed help on **CenterX** property

Object	Property description
Ellipse	Gets or sets the horizontal ellipse center position
StructFountainFillProperties	

CenterY Property

Select one of the available subtopics below to see detailed help on **CenterY** property

Object	Property description
Ellipse	Gets or sets the vertical ellipse center position
StructFountainFillProperties	

CloneParent Property

Select one of the available subtopics below to see detailed help on **CloneParent** property

Object	Property description
CloneLink	Gets the clone parent
Effect	Gets the parent effect for the cloned effects collection

Clones Property

Select one of the available subtopics below to see detailed help on **Clones** property

Object	Property description
Effect	Gets the collection of effects cloned from the current object
Shape	Gets the collection of cloned shapes

Closed Property

Select one of the available subtopics below to see detailed help on **Closed** property

Object	Property description
Curve	Gets or sets whether the curve is closed
SubPath	Gets or sets whether the subpath is open or closed

Color Property

Select one of the available subtopics below to see detailed help on **Color** property

Object	Property description
EffectDropShadow	Gets or sets the feather color
EffectLens	Gets or sets the lens color
FountainColor	Gets or sets the Color
Layer	Gets or sets the layer override color
Outline	Gets or sets an outline's color
Page	Gets or sets the page color
Palette	Gets or sets a color object based on index
SeparationPlate	property Color
TrapLayer	property Color

ColorAcceleration Property

Select one of the available subtopics below to see detailed help on **ColorAcceleration** property

Object	Property description
EffectBlend	Gets or sets whether or not the color accelerates along the path
EffectContour	Gets or sets the Color Acceleration

ColorBlendType Property

Select one of the available subtopics below to see detailed help on **ColorBlendType** property

Object	Property description
EffectBlend	Gets or sets the blend type of cdrFountainFillBlendType type
EffectContour	Gets or sets the color blend type

ColorCount Property

Select one of the available subtopics below to see detailed help on **ColorCount** property

Object	Property description
Palette	Gets the number of colors in a color palette
StructFountainFillProperties	

ColorMode Property

Select one of the available subtopics below to see detailed help on **ColorMode** property

Object	Property description
PDFVBASettings	
PrintOptions	property ColorMode

ColorResolution Property

Select one of the available subtopics below to see detailed help on **ColorResolution** property

Object	Property description
PDFVBASettings	
PrintOptions	property ColorResolution

CommandBars Property

Select one of the available subtopics below to see detailed help on **CommandBars** property

Object	Property description
Application	Gets a collection of all command bars
FrameWork	Gets a collection of all CommandBars in the FrameWork

Controls Property

Select one of the available subtopics below to see detailed help on **Controls** property

Object	Property description
CommandBar	Gets the collection of controls contained by the bar
CommandBarMode	property Controls

Count Property

Select one of the available subtopics below to see detailed help on **Count** property

Object	Property description
ArrowHeads	Gets the number of arrows heads in the collection
Colors	Gets the number of colors in the color collection
CommandBarControls	Gets the number of items in the collection
CommandBarModes	Gets the number of items in the collection
CommandBars	Gets the number of items in the collection
CrossPoints	Gets the number of CrossPoints in the CrossPoint collection
DataFields	Gets the number of items in the collection of DataFields
DataItems	Gets the number of items in the collection of dataitems
Documents	Gets the number of items in the collection of documents
Effects	Gets the number of items in the collection of effects
FontList	Gets the number of items in the list of fonts
FountainColors	Gets the number of FountainColor in the FountainColors collection
Layers	Gets the number of layers in the collection
NodeRange	Gets the number of nodes in the range
Nodes	Gets the number of nodes in the node collection
OutlineStyles	Gets the number of outlinestyles in the collection
Pages	Gets the number of items in the collection of pages
Palettes	Gets the number of items in the collection of palettes
PatternCanvases	Gets the number of PatternCanvas in the PatternCanvases collection
Points	Gets the number of points in the point collection
PrintDocuments	property Count
Printers	property Count
PrintPages	property Count
Properties	Gets the properties Count
RecentFiles	Gets the number of items in the collection of recent files
SegmentRange	Gets the number of segments in the range
Segments	Gets the number of segments in the segment collection
SeparationPlates	property Count
ShapeRange	Gets the number of shapes in the range
Shapes	Gets the number of shapes in the collection
Subpaths	Gets the number of subpaths in the subpath collection
TextureFillProperties	Gets the Number of Properties in the collection
TrapLayers	property Count
Views	Gets the number of items in the collection of views
Windows	Gets the number of items in the collection of windows
Workspaces	Gets the number of items in the collection of workspaces

CropMarks Property

Select one of the available subtopics below to see detailed help on **CropMarks** property

Object	Property description
PDFVBASettings	
PrintPrepress	property CropMarks

Default Property

Select one of the available subtopics below to see detailed help on **Default** property

Object	Property description
Printer	property Default
Printers	property Default
Workspace	Gets the workspace's default state

Description Property

Select one of the available subtopics below to see detailed help on **Description** property

Object	Property description
Printer	property Description
Workspace	Gets the workspace's description

Documents Property

Select one of the available subtopics below to see detailed help on **Documents** property

Object	Property description
Application	Gets the collection of documents open in the application
PrintJob	property Documents

DownsampleColor Property

Select one of the available subtopics below to see detailed help on **DownsampleColor** property

Object	Property description
PDFVBASettings	
PrintOptions	property DownsampleColor

DownsampleGray Property

Select one of the available subtopics below to see detailed help on **DownsampleGray** property

Object	Property description
PDFVBASettings	
PrintOptions	property DownsampleGray

DownsampleMono Property

Select one of the available subtopics below to see detailed help on **DownsampleMono** property

Object	Property description
PDFVBASettings	
PrintOptions	property DownsampleMono

EdgePad Property

Select one of the available subtopics below to see detailed help on **EdgePad** property

Object	Property description
FountainFill	Gets or sets a fountain fill's edge pad
StructFountainFillProperties	

Effects Property

Select one of the available subtopics below to see detailed help on **Effects** property

Object	Property description
EffectControlPath	Gets a collection of effects of type Blend and TextOnPath
ExtrudeVanishingPoint	Gets the collection of extrude objects sharing the same vanishing point
Shape	Gets collection of effects applied to object

Enabled Property

Select one of the available subtopics below to see detailed help on **Enabled** property

Object	Property description
CommandBar	Gets/sets the enabled status of the control
PrintSeparations	property Enabled
PrintTrapping	property Enabled
SeparationPlate	property Enabled

EndNode Property

Select one of the available subtopics below to see detailed help on **EndNode** property

Object	Property description
Segment	Gets the end node
SubPath	Gets the end node

FileName Property

Select one of the available subtopics below to see detailed help on **FileName** property

Object	Property description
CorelScriptFile	Gets the Corel SCRIPT file name
Document	File Name
PrintSettings	property FileName

FilePath Property

Select one of the available subtopics below to see detailed help on **FilePath** property

Object	Property description
Document	File Path
PatternFill	Gets the FilePath

Fill Property

Select one of the available subtopics below to see detailed help on **Fill** property

Object	Property description
Shape	Gets or sets the fill properties
StructFontProperties	

FillColor Property

Select one of the available subtopics below to see detailed help on **FillColor** property

Object	Property description
EffectContour	Gets or sets the contour fill color
EffectLens	Gets or sets the lens fill color

Filter Property

Select one of the available subtopics below to see detailed help on **Filter** property

Object	Property description
AddinHook	
StructSaveAsOptions	

Fountain Property

Select one of the available subtopics below to see detailed help on **Fountain** property

Object	Property description
Fill	Gets or sets Fountain properties
Transparency	Gets or sets the fountain fill properties of the transparency

FountainSteps Property

Select one of the available subtopics below to see detailed help on **FountainSteps** property

Object	Property description
PDFVBASettings	
PrintOptions	property FountainSteps

Frequency Property

Select one of the available subtopics below to see detailed help on **Frequency** property

Object	Property description
EffectZipperDistortion	Gets or sets the frequency of the Zipper distortion
SeparationPlate	property Frequency

FromColor Property

Select one of the available subtopics below to see detailed help on **FromColor** property

Object	Property description
EffectLens	Gets or sets the starting color for custom color map lens
StructFountainFillProperties	

Frozen Property

Select one of the available subtopics below to see detailed help on **Frozen** property

Object	Property description
EffectLens	Gets whether or not the lens is frozen
Transparency	Gets whether or not the transparency is frozen

GrayResolution Property

Select one of the available subtopics below to see detailed help on **GrayResolution** property

Object	Property description
PDFVBASettings	
PrintOptions	property GrayResolution

Height Property

Select one of the available subtopics below to see detailed help on **Height** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window height
CommandBar	Gets the height of the control
CommandBarControl	Gets/sets the height of the control
PatternCanvas	Gets or sets the Height
Window	Gets or sets the window's height

Index Property

Select one of the available subtopics below to see detailed help on **Index** property

Object	Property description
AddinHook	
ArrowHead	ArrowHead Index
CommandBar	Gets/sets the index of the control in it's collection
DataField	Gets Index
Document	Gets the page index
Node	Gets the index of the node within its subpath
OutlineStyle	Gets the index
Page	Gets the page number
PatternCanvas	Gets the Index
PostScriptFill	Gets the PostScript Fill's Index
Properties	Index of the property
RecentFile	Gets the recent file's position in the Recent File list
Segment	Gets the index of the segment within its subpath
SubPath	Gets the index of the subpath within its curve
Window	Gets the index into the windows collection

Item Property

Select one of the available subtopics below to see detailed help on **Item** property

Object	Property description
ArrowHeads	Gets a reference to the specified arrow head, in base 1
Colors	Gets a reference to the specified color, in base 1
CommandBarControls	Gets the item in the collection at the specified index
CommandBarModes	Gets the item in the collection at the specified index
CommandBars	Gets the item in the collection at the specified index
CrossPoints	Gets a reference to the specified CrossPoint, in base 1
DataFields	Gets a reference to a specified DataField, in base 1
DataItems	Gets a reference to a specified dataitem, in base 1
Documents	Gets a reference to a specified document, in base 1
Effects	Gets a reference to the effect specified, in base 1
FontList	Gets a reference to a specified font, in base 1
FountainColors	Gets or sets a reference to the specified FountainColor, in base 1
Layers	Gets a reference to the layer specified, in base 1
NodeRange	Gets a reference to the specified node, in base 1
Nodes	Gets a reference to the specified node, in base 1
OutlineStyles	Gets a reference to the specified outlinestyle, in base 1
Pages	Gets a reference to the specified page, in base 1; however, using 0 brings you the Master Page
Palettes	Gets a reference to a specified palette, in base 1
PatternCanvases	Gets or sets a reference to the specified PatternCanvas, in base 1
Points	Gets a reference to the specified point, in base 1
Printers	property Item
Properties	Gets or sets an Item
RecentFiles	Gets a reference to a specified window, in base 1
SegmentRange	Gets a reference to the specified segment, in base 1
Segments	Gets a reference to the specified segment, in base 1
SeparationPlates	property Item
ShapeRange	Gets a reference to the shape specified, in base 1
Shapes	Gets a reference to the shape specified, in base 1
Subpaths	Gets a reference to the specified node, in base 1
TextureFillProperties	Gets a reference to a Texture Property, in base 1
TrapLayers	property Item
Views	Gets a reference to a specified view, in base 1
Windows	Gets a reference to a specified window, in base 1
Workspaces	Gets a reference to a specified workspace, in base 1

Keywords Property

Select one of the available subtopics below to see detailed help on **Keywords** property

Object	Property description
Document	Gets or sets document keywords
PDFVBASettings	

Layers Property

Select one of the available subtopics below to see detailed help on **Layers** property

Object	Property description
Page	Gets the collection of layers on the page
PrintTrapping	property Layers

Left Property

Select one of the available subtopics below to see detailed help on **Left** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window left position
CommandBar	Gets/sets the screen position of the control's left-most edge
Window	Gets or sets the window's left value

Length Property

Select one of the available subtopics below to see detailed help on **Length** property

Object	Property description
Curve	Gets the length of the curve
Segment	Gets the length of the segment (in the document's units)
SegmentRange	Gets the total length of the segments within the range
SubPath	Gets the length of the subpath (in document's units)

LinkAcceleration Property

Select one of the available subtopics below to see detailed help on **LinkAcceleration** property

Object	Property description
EffectBlend	Gets or sets whether or not the link accelerates along the path
EffectContour	Gets or sets the Link Acceleration

MainMenu Property

Select one of the available subtopics below to see detailed help on **MainMenu** property

Object	Property description
Application	Gets the main menu
FrameWork	Gets the main menu of the application

MaintainOPILinks Property

Select one of the available subtopics below to see detailed help on **MaintainOPILinks** property

Object	Property description
PDFVBASettings	
PrintPostScript	property MaintainOPILinks

MidPoint Property

Select one of the available subtopics below to see detailed help on **MidPoint** property

Object	Property description
FountainFill	Gets or sets a fountain fill's midpoint position
StructFountainFillProperties	

MirrorFill Property

Select one of the available subtopics below to see detailed help on **MirrorFill** property

Object	Property description
PatternFill	Gets or sets Mirror Fill
TextureFill	Gets or sets Mirror Fill

Mode Property

Select one of the available subtopics below to see detailed help on **Mode** property

Object	Property description
EffectBlend	Gets or sets the blend mode of cdrBlendMode type
EffectEnvelope	Gets or sets the mode of type cdrEnvelopeMode for the envelope

MonoResolution Property

Select one of the available subtopics below to see detailed help on **MonoResolution** property

Object	Property description
PDFVBASettings	
PrintOptions	property MonoResolution

Name Property

Select one of the available subtopics below to see detailed help on **Name** property

Object	Property description
Color	Gets color name
CommandBar	Gets/sets the name of the control
CommandBarMode	Gets/sets the name of the control
CorelScriptFile	Gets or sets a friendly name for the script; will be used as a macro name if the script is translated to VBA
DataField	Gets or sets the datafield Name
Document	Gets or sets the document's name
FrameWork	Gets/sets the name of the control
Layer	Gets or sets the layer name
Page	Gets or sets the page's name
Palette	Gets or sets the color palette name; palette name can only be set if it is a custom palette
PostScriptFill	Gets the PostScript Fill's Name
Printer	property Name
RecentFile	Gets or sets the recent file's name
Shape	Gets or sets a shape's name
StructFontProperties	
TextureFillProperty	Gets the Name of the Property
View	Gets or sets the name of the view
Workspace	Gets the workspace's name

NameLocal Property

Select one of the available subtopics below to see detailed help on **NameLocal** property

Object	Property description
CommandBar	Gets/sets the localized name of the control
CommandBarMode	Gets/sets the localized name of the control

Next Property

Select one of the available subtopics below to see detailed help on **Next** property

Object	Property description
Shape	Gets the next shape
Window	Gets the next window

Nodes Property

Select one of the available subtopics below to see detailed help on **Nodes** property

Object	Property description
Curve	Gets the collection of nodes within the curve
SubPath	Gets the collection of nodes within the subpath

Offset Property

Select one of the available subtopics below to see detailed help on **Offset** property

Object	Property description
CrossPoint	CrossPoint Offset from the object's first node
EffectContour	Gets or sets the contour offset
EffectTextOnPath	Gets or sets the text offset from path

Orientation Property

Select one of the available subtopics below to see detailed help on **Orientation** property

Object	Property description
EffectTextOnPath	Gets or sets the text orientation of type cdrFittedOrientation
Page	Gets or sets the page's orientation

OriginX Property

Select one of the available subtopics below to see detailed help on **OriginX** property

Object	Property description
ActiveView	Gets or sets the origin x of the active view
EffectDistortion	Gets or sets the X-value of the origin
PatternFill	Gets or sets the horizontal Origin
TextureFill	Gets or sets the horizontal Origin
View	Gets or sets the origin x of the view

OriginY Property

Select one of the available subtopics below to see detailed help on **OriginY** property

Object	Property description
ActiveView	Gets or sets the origin y of the active view
EffectDistortion	Gets or sets the Y-value of the origin
PatternFill	Gets or sets the vertical Origin
TextureFill	Gets or sets the vertical Origin
View	Gets or sets the origin y of the view

Outline Property

Select one of the available subtopics below to see detailed help on **Outline** property

Object	Property description
Shape	Allows users to access the outline properties
StructFontProperties	

OutlineColor Property

Select one of the available subtopics below to see detailed help on **OutlineColor** property

Object	Property description
EffectContour	Gets or sets the contour outline color
EffectLens	Gets or sets the lens outline color

Overwrite Property

Select one of the available subtopics below to see detailed help on **Overwrite** property

Object	Property description
StructExportOptions	
StructSaveAsOptions	

Page Property

Select one of the available subtopics below to see detailed help on **Page** property

Object	Property description
View	Gets or sets the page of the view
Window	Gets the active page for that window

PageRange Property

Select one of the available subtopics below to see detailed help on **PageRange** property

Object	Property description
PDFVBASettings	
PrintSettings	property PageRange

Pages Property

Select one of the available subtopics below to see detailed help on **Pages** property

Object	Property description
Document	Returns a collection of pages for the document
PrintJob	property Pages

PaletteID Property

Select one of the available subtopics below to see detailed help on **PaletteID** property

Object	Property description
Color	Gets the Fixed Palette Color's ID
Palette	Gets the palette ID

Parent Property

Select one of the available subtopics below to see detailed help on **Parent** property

Object	Property description
ActiveView	Gets the parent of which this object is a child
AddinHook	
AddIns	
Application	Gets the application to which the object belongs
AppWindow	Gets the parent of which the object is a child
ArrowHeads	Gets the parent of which the arrowhead collection is a child
Clipboard	Gets the parent of which this object is a child
CloneLink	Gets the parent of which this object is a child
Color	Gets the parent of which this object is a child
Colors	Gets the parent of which the color collection is a child; if the color object is not from a palette the color parent is null
CorelScriptFile	Gets the parent of which the object is a child
DataField	Gets the parent of which this object is a child
DataFields	Gets the parent of which this object is a child
DataItem	Gets the parent of which this object is a child
DataItems	Gets the parent of which this object is a child
Document	Gets the parent object of which the document object is a child
Documents	Gets the parent of which the document collection is a child
Effect	Gets the parent of which this object is a child
EffectArtistic	Gets the parent of which this object is a child
EffectBlend	Gets the application to which the object belongs
EffectContour	Gets the parent of which this object is a child
EffectControlPath	Gets the parent of which this object is a child
EffectDistortion	Gets the parent of which this object is a child
EffectDropShadow	Gets the parent of which this object is a child
EffectEnvelope	Gets the parent of which this object is a child
EffectExtrude	Gets the parent of which this object is a child
EffectLens	Gets the parent of which this object is a child
EffectPerspective	Gets the parent of which this object is a child
EffectPushPullDistortion	Gets the parent of which this object is a child
Effects	Gets the parent of which this object is a child
EffectTextOnPath	Gets the parent of which this object is a child
EffectTwisterDistortion	Gets the parent of which this object is a child
EffectZipperDistortion	Gets the parent of which this object is a child
ExtrudeVanishingPoint	Gets the parent of which this object is a child
FontList	Gets the parent of which the font list is a child
Grid	Gets the parent of which this object is a child
Layer	Gets the parent object of the layer

Layers	Gets the parent of which the layer collection is a child
Node	Gets the node collection parent
NodeRange	Gets the parent shape
Nodes	Gets the parent (shape or subpath)
OutlineStyles	Gets the parent of which the outlinestyle collection is a child
Page	Gets the parent of which this object is a child
Pages	Gets the parent of which the page collection is a child
Palette	Gets the parent of which this object is a child
Palettes	Gets the parent of which the palette collection is a child
Points	Gets the shape parent
PowerClip	Gets the parent of which this object is a child
RecentFile	Gets the parent of which this object is a child
RecentFiles	Gets the parent of which this object is a child
Rulers	Gets the parent of which this object is a child
Segment	Gets the segment collection parent
SegmentRange	Gets the parent shape
Segments	Gets the parent subpath
Shape	Gets the parent of which the object is a child: if the shape is a Selection Shape it will be a document, otherwise it will be a layer
ShapePoint	Gets the shape parent
ShapeRange	Gets the parent of which the shape range is a child
Shapes	Gets the parent of which the shape collection is a child
SubPath	Gets the subpath collection parent
Subpaths	Gets the parent shape
Transparency	Gets the parent of which this object is a child
View	Gets the parent of which this object is a child
Views	Gets the parent of which this object is a child
Window	Gets the parent of which this object is a child
Windows	Gets the parent of which the window collection is a child
Workspace	Gets the parent of which this object is a child
Workspaces	Gets the parent of which the workspace collection is a child

Path Property

Select one of the available subtopics below to see detailed help on **Path** property

Object	Property description
Application	Gets the full path of the 'Draw' directory
EffectBlend	Gets or sets the blend path
EffectTextOnPath	Gets or sets the path object
RecentFile	Gets or sets the recent file's path

Pattern Property

Select one of the available subtopics below to see detailed help on **Pattern** property

Object	Property description
Fill	Gets or sets Pattern properties
Transparency	Gets or sets the pattern fill properties of the transparency

Position Property

Select one of the available subtopics below to see detailed help on **Position** property

Object	Property description
CommandBar	Gets/sets the cuiBarPosition of the bar
FountainColor	Gets the Position
StructFontProperties	

PositionX Property

Select one of the available subtopics below to see detailed help on **PositionX** property

Object	Property description
CrossPoint	CrossPoint X coordinate
ExtrudeVanishingPoint	Gets or sets the vanishing point X position
Node	Gets or sets the horizontal position (X) of the node on the page, according to the document's reference point
NodeRange	Gets the X position of the leftmost node within the range
Shape	Gets or sets the horizontal position (X) of the shape on the page, according to the document's reference point
ShapePoint	Gets or sets the Point's X coordinate
ShapeRange	Gets or sets the horizontal position (X) of the shape on the page, according to the document's reference point
SubPath	Gets or sets the X position of the subpath's head

PositionY Property

Select one of the available subtopics below to see detailed help on **PositionY** property

Object	Property description
CrossPoint	CrossPoint Y coordinate
ExtrudeVanishingPoint	Gets or sets the vanishing point Y position
Node	Gets or sets the vertical position (Y) of the node on the page, according to the document's reference point
NodeRange	Gets the Y position of the topmost node within the range
Shape	Gets or sets the vertical position (Y) of the shape on the page, according to the document's reference point
ShapePoint	Gets or sets the Point's Y coordinate
ShapeRange	Gets or sets the vertical position (Y) of the shape on the page, according to the document's reference point
SubPath	Gets or sets the Y position of the subpath's head

PostScript Property

Select one of the available subtopics below to see detailed help on **PostScript** property

Object	Property description
Fill	Gets or sets PostScript properties
PrintSettings	property PostScript

Previous Property

Select one of the available subtopics below to see detailed help on **Previous** property

Object	Property description
Shape	Gets the previous shape
Window	Gets the previous window

Properties Property

Select one of the available subtopics below to see detailed help on **Properties** property

Object	Property description
Document	Gets Properties
Layer	Gets Properties
Page	Gets Properties
PostScriptFill	Gets or sets PostScript Fill Properties
Shape	Gets Properties
TextureFill	Gets Texture Fill properties

RegistrationMarks Property

Select one of the available subtopics below to see detailed help on **RegistrationMarks** property

Object	Property description
PDFVBASettings	
PrintPrepress	property RegistrationMarks

Resolution Property

Select one of the available subtopics below to see detailed help on **Resolution** property

Object	Property description
Document	Gets or sets the Document's Resolution in dots per inch
Page	Gets or sets the page's resolution in dots per inch (dpi)
PrintSeparations	property Resolution
TextureFill	Gets or sets Resolution

ResolutionX Property

Select one of the available subtopics below to see detailed help on **ResolutionX** property

Object	Property description
Bitmap	Gets the bitmap's resolution (X)
StructExportOptions	

ResolutionY Property

Select one of the available subtopics below to see detailed help on **ResolutionY** property

Object	Property description
Bitmap	Gets the bitmap's resolution (Y)
StructExportOptions	

RotationAngle Property

Select one of the available subtopics below to see detailed help on **RotationAngle** property

Object	Property description
PatternFill	Gets or sets the Rotation angle
Shape	Gets or sets the rotation angle of the shape on the page
TextureFill	Gets or sets the Rotation angle

RotationCenterX Property

Select one of the available subtopics below to see detailed help on **RotationCenterX** property

Object	Property description
Shape	Gets or sets the horizontal center of rotation (X) for the shape on the page
ShapeRange	Gets or sets the horizontal center of rotation (X) for the shape on the page

RotationCenterY Property

Select one of the available subtopics below to see detailed help on **RotationCenterY** property

Object	Property description
Shape	Gets or sets the vertical center of rotation (Y) for the shape on the page
ShapeRange	Gets or sets the vertical center of rotation (Y) for the shape on the page

Segments Property

Select one of the available subtopics below to see detailed help on **Segments** property

Object	Property description
Curve	Gets the collection of segments within the curve
SubPath	Gets the collection of segments within the subpath

Shapes Property

Select one of the available subtopics below to see detailed help on **Shapes** property

Object	Property description
EffectLens	Gets the subshapes of frozen lens group
Layer	Gets a collection of shapes on the layer
Page	Gets the collection of shapes on the page, across all layers
PowerClip	Gets the shapes in the powerclip
Shape	Gets a collection of shapes

Size Property

Select one of the available subtopics below to see detailed help on **Size** property

Object	Property description
PatternCanvas	Gets or sets the Size
StructFontProperties	

SizeHeight Property

Select one of the available subtopics below to see detailed help on **SizeHeight** property

Object	Property description
Bitmap	Gets the bitmap's vertical size (height)
NodeRange	Gets the height between the upmost & downmost node
Page	Gets or sets the page height size
Shape	Gets or sets the height of the shape on the page
ShapeRange	Gets or sets the height of the shape on the page
SubPath	Gets the height of the whole subpath (in document's units)

SizeWidth Property

Select one of the available subtopics below to see detailed help on **SizeWidth** property

Object	Property description
Bitmap	Gets the bitmap's horizontal size (width)
NodeRange	Gets the width between the leftmost & rightmost node within the range
Page	Gets or sets the page width size
Shape	Gets or sets the width of the shape on the page
ShapeRange	Gets or sets the width of the shape on the page
SubPath	Gets the width of the whole subpath (in document's units)

SkewAngle Property

Select one of the available subtopics below to see detailed help on **SkewAngle** property

Object	Property description
PatternFill	Gets or sets the Skew angle
TextureFill	Gets or sets the Skew angle

SpacingAcceleration Property

Select one of the available subtopics below to see detailed help on **SpacingAcceleration** property

Object	Property description
EffectBlend	Gets or sets whether or not the spacing accelerates along the path
EffectContour	Gets or sets the Spacing Acceleration

StartNode Property

Select one of the available subtopics below to see detailed help on **StartNode** property

Object	Property description
Segment	Gets the start node
SubPath	Gets the start node

StatusBar Property

Select one of the available subtopics below to see detailed help on **StatusBar** property

Object	Property description
Application	Gets the status bar
FrameWork	Gets the main statusbar of the application

Steps Property

Select one of the available subtopics below to see detailed help on **Steps** property

Object	Property description
EffectBlend	Gets or sets the number of steps in the blend
EffectContour	Gets or sets the number of steps in the contour
FountainFill	Gets or sets a fountain fill's steps
StructFountainFillProperties	

Style Property

Select one of the available subtopics below to see detailed help on **Style** property

Object	Property description
Outline	Gets or sets an outline's style
StructFontProperties	

SubPath Property

Select one of the available subtopics below to see detailed help on **SubPath** property

Object	Property description
Node	Gets the subpath which holds this node
Segment	Gets the subpath to which the segment belongs

SubPathIndex Property

Select one of the available subtopics below to see detailed help on **SubPathIndex** property

Object	Property description
Node	Gets the index of the node's subpath within the curve
Segment	Gets the index of the segment's subpath within the curve

Text Property

Select one of the available subtopics below to see detailed help on **Text** property

Object	Property description
EffectTextOnPath	Gets or sets the text object
Shape	Allows users to access the text properties

Texture Property

Select one of the available subtopics below to see detailed help on **Texture** property

Object	Property description
Fill	Gets or sets Texture properties
Transparency	Gets or sets the texture fill properties of the transparency

TileHeight Property

Select one of the available subtopics below to see detailed help on **TileHeight** property

Object	Property description
PatternFill	Gets or sets the Tile Height
TextureFill	Gets or sets the Tile Height

TileOffset Property

Select one of the available subtopics below to see detailed help on **TileOffset** property

Object	Property description
PatternFill	Gets or sets the Tile offset
TextureFill	Gets or sets the Tile offset

TileOffsetType Property

Select one of the available subtopics below to see detailed help on **TileOffsetType** property

Object	Property description
PatternFill	Gets or sets the Tile offset Type
TextureFill	Gets or sets the Tile offset Type

TileWidth Property

Select one of the available subtopics below to see detailed help on **TileWidth** property

Object	Property description
PatternFill	Gets or sets the Tile Width
TextureFill	Gets or sets the Tile Width

ToColor Property

Select one of the available subtopics below to see detailed help on **ToColor** property

Object	Property description
EffectLens	Gets or sets the ending color for custom color map lens
StructFountainFillProperties	

Top Property

Select one of the available subtopics below to see detailed help on **Top** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window top position
CommandBar	Gets/sets the screen position of the control's top-most edge
Window	Gets or sets the window's top value

TransformWithShape Property

Select one of the available subtopics below to see detailed help on **TransformWithShape** property

Object	Property description
PatternFill	Gets or sets the Transform With Shape value
TextureFill	Gets or sets the Transform With Shape value

TrueTypeToType1 Property

Select one of the available subtopics below to see detailed help on **TrueTypeToType1** property

Object	Property description
PDFVBASettings	
PrintPostScript	property TrueTypeToType1

Type Property

Select one of the available subtopics below to see detailed help on **Type** property

Object	Property description
ActiveView	Gets or sets the type of the active view
Color	Gets color type
CommandBar	Gets the cuiBarType of the bar
Effect	Gets the effect with a cdrEffectType type
EffectDistortion	Gets or sets the type of distortion of cdrDistortionType type
EffectDropShadow	Gets or sets the feather type with a cdrDSType type
EffectExtrude	Gets or sets the extrude type of cdrExtrudeType type
EffectLens	Gets or sets the lens type of cdrLensType type
Ellipse	Gets or sets the type of ellipse
ExtrudeVanishingPoint	Gets or sets the vanishing point type of cdrVPType type
Fill	Gets fill type
FountainFill	Gets or sets a fountain fill's type
Grid	Gets or sets the grid type
Guide	Gets the type of guideline
Node	Gets or sets the node type
NodeRange	Gets the node type, if common within the range
Outline	Gets or sets an outline's type
Palette	Gets whether or not a color palette is custom or fixed
PatternFill	Gets or sets the Type
Polygon	Gets or sets the type of polygon
Printer	property Type
Segment	Gets or sets the segment type
SegmentRange	Gets the segment type, if common within the range
SeparationPlate	property Type
Shape	Gets the shape's type
ShapePoint	Gets the Point Type
StructFountainFillProperties	
Text	Gets text type: Artistic or Paragraph
TextureFillProperty	Gets the Type of the Property
Transparency	Gets the transparency type
TrapLayer	property Type

UseColorProfile Property

Select one of the available subtopics below to see detailed help on **UseColorProfile** property

Object	Property description
PDFVBASettings	
PrintOptions	property UseColorProfile
StructExportOptions	

Value Property

Select one of the available subtopics below to see detailed help on **Value** property

Object	Property description
DataItem	Gets or sets the dataitem Value
TextureFillProperty	Gets or sets the Value of the Property

Version Property

Select one of the available subtopics below to see detailed help on **Version** property

Object	Property description
Application	Gets a verbose description of the current DRAW version
StructSaveAsOptions	

Visible Property

Select one of the available subtopics below to see detailed help on **Visible** property

Object	Property description
Application	Gets or sets the visibility of the application window; allows you to hide or show the application
CommandBar	Gets/sets the visible status of the control
CommandBarControl	Gets/sets the visible status of the control
Grid	Gets or sets the grid's visibility state
Layer	Gets or sets the layer's visibility state

Width Property

Select one of the available subtopics below to see detailed help on **Width** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window width
CommandBar	Gets the width of the control
CommandBarControl	Gets/sets the width of the control
Outline	Gets or sets an outline's width
PatternCanvas	Gets or sets the Width
PrintTrapping	property Width
Window	Gets or sets the window's width

Windows Property

Select one of the available subtopics below to see detailed help on **Windows** property

Object	Property description
Application	Gets a collection of windows open in the application
Document	Returns a collection of windows for the document

WindowState Property

Select one of the available subtopics below to see detailed help on **WindowState** property

Object	Property description
AppWindow	Gets or sets the CorelDRAW's window state
Window	Gets or sets the window's state

Zoom Property

Select one of the available subtopics below to see detailed help on **Zoom** property

Object	Property description
ActiveView	Gets or sets the zoom level of the active view
View	Gets or sets the zoom level of the active view

Activate Method

Select one of the available subtopics below to see detailed help on **Activate** method

Object	Method description
AppWindow	Activates CorelDRAW's window
Document	Activates the current document
Layer	Activates the layer
Page	Activates the page
View	Activates the view
Window	Activates the window
Workspace	Activates the workspace

Add Method

Select one of the available subtopics below to see detailed help on **Add** method

Object	Method description
CommandBars	Adds an object to the collection
DataFields	Adds a field
DataItems	Adds a field
FountainColors	Adds a FountainColor to the FountainColors collection
NodeRange	Adds a node to the range
OutlineStyles	Adds a new outline style
PatternCanvases	Adds a PatternCanvas to the PatternCanvases collection
RecentFiles	Adds a file to the Recent Files list
SegmentRange	Adds a segment to the range
ShapeRange	Adds a shape in a range

AddNode Method

Select one of the available subtopics below to see detailed help on **AddNode** method

Object	Method description
CorelScript	
SegmentRange	Adds a node at the middle of each segment within the range

AddNodeAt Method

Select one of the available subtopics below to see detailed help on **AddNodeAt** method

Object	Method description
Segment	Inserts a node on the segment
SubPath	Inserts a node on the segment

AddRange Method

Select one of the available subtopics below to see detailed help on **AddRange** method

Object	Method description
NodeRange	Adds all nodes from the given range to the current range
SegmentRange	Adds all segments from the given range to the current range
ShapeRange	Adds a shape range in a range

AddToPowerClip Method

Select one of the available subtopics below to see detailed help on **AddToPowerClip** method

Object	Method description
Shape	Adds this shape to a power clip
ShapeRange	Adds this shape to a power clip

All Method

Select one of the available subtopics below to see detailed help on **All** method

Object	Method description
Nodes	Creates a range containing all the nodes of the collection
Segments	Creates a range containing all the segments of the collection
Shapes	Gets the shape range

AllExcluding Method

Select one of the available subtopics below to see detailed help on **AllExcluding** method

Object	Method description
Nodes	Creates a range containing all the nodes except the ones specified
Segments	Creates a range containing all the segments except the ones specified
Shapes	Gets the shape range excluding the indexes

ApplyFountainFill Method

Select one of the available subtopics below to see detailed help on **ApplyFountainFill** method

Object	Method description
CorelScript	
Fill	Apply a Fountain Fill to the shape
ShapeRange	Apply a Fountain Fill to the shape

ApplyNoFill Method

Select one of the available subtopics below to see detailed help on **ApplyNoFill** method

Object	Method description
CorelScript	
Fill	Apply no Fill to the shape
ShapeRange	Apply no Fill to the shape

ApplyPatternFill Method

Select one of the available subtopics below to see detailed help on **ApplyPatternFill** method

Object	Method description
Fill	Apply a Pattern Fill to the shape
ShapeRange	Apply a Pattern Fill to the shape

ApplyPostscriptFill Method

Select one of the available subtopics below to see detailed help on **ApplyPostscriptFill** method

Object	Method description
CorelScript	
Fill	Apply a PostScript Fill to the shape
ShapeRange	Apply a PostScript Fill to the shape

ApplyTextureFill Method

Select one of the available subtopics below to see detailed help on **ApplyTextureFill** method

Object	Method description
CorelScript	
Fill	Apply a Texture Fill to the shape
ShapeRange	Apply a Texture Fill to the shape

ApplyUniformFill Method

Select one of the available subtopics below to see detailed help on **ApplyUniformFill** method

Object	Method description
Fill	Apply a Uniform Fill to the shape
ShapeRange	Apply a Uniform Fill to the shape

BeginCommandGroup Method

Select one of the available subtopics below to see detailed help on **BeginCommandGroup** method

Object	Method description
CorelScript	
Document	BeginCommandGroup

BreakApart Method

Select one of the available subtopics below to see detailed help on **BreakApart** method

Object	Method description
CorelScript	
Node	Breaks this node into two unconnected nodes
NodeRange	Breaks apart the curve at each node in the range
Shape	Breaks Apart

BreakApartAt Method

Select one of the available subtopics below to see detailed help on **BreakApartAt** method

Object	Method description
Segment	Breaks this segment into two unconnected segments
SubPath	Breaks this segment into two unconnected segments

Clear Method

Select one of the available subtopics below to see detailed help on **Clear** method

Object	Method description
Clipboard	Clears the contents of the clipboard
DataItem	Clears the dataitem
DataItems	Clears DataItem collection
Effect	Clears the blend effect for the object
PatternCanvas	Clears the Pattern Canvas
PrintJob	method Clear

ClearEffect Method

Select one of the available subtopics below to see detailed help on **ClearEffect** method

Object	Method description
CorelScript	
Shape	Clears Effect

Clone Method

Select one of the available subtopics below to see detailed help on **Clone** method

Object	Method description
Shape	Clones the shape
ShapeRange	Clone

Close Method

Select one of the available subtopics below to see detailed help on **Close** method

Object	Method description
Document	Closes the document
Palette	Closes the color palette
Window	Closes the window

Combine Method

Select one of the available subtopics below to see detailed help on **Combine** method

Object	Method description
CorelScript	
Shape	Combines
ShapeRange	Combines

ConvertOutlineToObject Method

Select one of the available subtopics below to see detailed help on **ConvertOutlineToObject** method

Object	Method description
CorelScript	
ShapeRange	Converts an outline into an object

ConvertToBitmap Method

Select one of the available subtopics below to see detailed help on **ConvertToBitmap** method

Object	Method description
CorelScript	
Shape	Converts the shape from a vector to a raster image
ShapeRange	Converts to bitmap

ConvertToBitmapEx Method

Select one of the available subtopics below to see detailed help on **ConvertToBitmapEx** method

Object	Method description
Shape	Converts to bitmap ex
ShapeRange	Converts to bitmap

ConvertToCurves Method

Select one of the available subtopics below to see detailed help on **ConvertToCurves** method

Object	Method description
CorelScript	
Shape	Converts a shape to curves
ShapeRange	Convert to curves

Copy Method

Select one of the available subtopics below to see detailed help on **Copy** method

Object	Method description
Shape	Copies the shape
ShapeRange	Copy

CopyFrom Method

Select one of the available subtopics below to see detailed help on **CopyFrom** method

Object	Method description
DataItems	Copies DataItems from a Shape
EffectBlend	Copies the blend from another object
EffectEnvelope	Copies the envelope properties from another envelope effect
EffectExtrude	Copies the extrude properties from another extrude object

CreateArtisticText Method

Select one of the available subtopics below to see detailed help on **CreateArtisticText** method

Object	Method description
CorelScript	
Layer	Creates artistic text on a layer

CreateConnector Method

Select one of the available subtopics below to see detailed help on **CreateConnector** method

Object	Method description
CorelScript	
Layer	Creates a connector on a layer

CreateEllipse Method

Select one of the available subtopics below to see detailed help on **CreateEllipse** method

Object	Method description
CorelScript	
Layer	Creates an ellipse on a layer

CreateGridBoxes Method

Select one of the available subtopics below to see detailed help on **CreateGridBoxes** method

Object	Method description
CorelScript	
Layer	Creates grid boxes on a layer

CreateRectangle Method

Select one of the available subtopics below to see detailed help on **CreateRectangle** method

Object	Method description
CorelScript	
Layer	Creates a rectangle on a layer

CreateSelection Method

Select one of the available subtopics below to see detailed help on **CreateSelection** method

Object	Method description
Shape	Makes a selection from a shape
ShapeRange	Makes a selection from a shape range

CreateSpiral Method

Select one of the available subtopics below to see detailed help on **CreateSpiral** method

Object	Method description
CorelScript	
Layer	Creates a spiral on a layer

Cut Method

Select one of the available subtopics below to see detailed help on **Cut** method

Object	Method description
Shape	Cuts the shape
ShapeRange	Cut

Delete Method

Select one of the available subtopics below to see detailed help on **Delete** method

Object	Method description
CommandBar	Deletes the control
CorelScriptFile	Deletes the Corel SCRIPT file from disk
DataField	Deletes datafield
FountainColor	Removes the object from the FountainColors collection
Layer	Deletes the layer
Node	Removes this node from the curve
NodeRange	Deletes all the nodes within the range
Page	Deletes the page
Properties	Deletes a property
RecentFile	Removes the file from the Recent File list
Shape	Deletes the shape
ShapeRange	Deletes a shape in a range
SubPath	Erases the whole subpath
View	Deletes the view

Duplicate Method

Select one of the available subtopics below to see detailed help on **Duplicate** method

Object	Method description
Shape	Duplicates the shape
ShapeRange	Duplicates

EndCommandGroup Method

Select one of the available subtopics below to see detailed help on **EndCommandGroup** method

Object	Method description
CorelScript	
Document	EndCommandGroup

FindShape Method

Select one of the available subtopics below to see detailed help on **FindShape** method

Object	Method description
Layer	Find Shape
Page	Find Shape

FindShapes Method

Select one of the available subtopics below to see detailed help on **FindShapes** method

Object	Method description
Layer	Find Shapes
Page	Find Shapes

Flip Method

Select one of the available subtopics below to see detailed help on **Flip** method

Object	Method description
Shape	Flips
ShapeRange	Flips

Freeze Method

Select one of the available subtopics below to see detailed help on **Freeze** method

Object	Method description
EffectLens	Freezes the lens effect
Transparency	Freezes the transparency

GetBoundingBox Method

Select one of the available subtopics below to see detailed help on **GetBoundingBox** method

Object	Method description
Shape	Gets the shape bounding box relatively to its lower left corner
ShapeRange	Gets the shape range bounding box relatively to its lower left corner

GetColor Method

Select one of the available subtopics below to see detailed help on **GetColor** method

Object	Method description
CorelScript	
CorelScriptTools	Gets Color

GetCurvatureAt Method

Select one of the available subtopics below to see detailed help on **GetCurvatureAt** method

Object	Method description
Segment	Gets the Curve's Curvature at the current segment
SubPath	Gets the Curve's Curvature at the current subpath

GetCurveSpeedAt Method

Select one of the available subtopics below to see detailed help on **GetCurveSpeedAt** method

Object	Method description
Segment	Gets the Curve's Speed at the current segment
SubPath	Gets the Curve's Speed at the current subpath

GetIntersections Method

Select one of the available subtopics below to see detailed help on **GetIntersections** method

Object	Method description
Segment	Finds the intersection point(s) with another segment
SubPath	Finds the intersection point(s) with another subpath

GetPerpendicularAt Method

Select one of the available subtopics below to see detailed help on **GetPerpendicularAt** method

Object	Method description
Segment	Finds the perpendicular line to a point on the segment
SubPath	Finds the perpendicular line to a point on the segment

GetPointPositionAt Method

Select one of the available subtopics below to see detailed help on **GetPointPositionAt** method

Object	Method description
Segment	Gets the coordinates of a point located on the curve
SubPath	Gets the coordinates of a point located on the subpath

GetPosition Method

Select one of the available subtopics below to see detailed help on **GetPosition** method

Object	Method description
CorelScript	
Node	Gets the horizontal and vertical position of the node on the page
Shape	Gets Position
ShapeRange	Gets Position
SubPath	Gets the horizontal and vertical position of the subpath's head on the page

GetSize Method

Select one of the available subtopics below to see detailed help on **GetSize** method

Object	Method description
CorelScript	
Page	Gets the size of the page
Shape	Gets Size
ShapeRange	Gets Size

GetTangentAt Method

Select one of the available subtopics below to see detailed help on **GetTangentAt** method

Object	Method description
Segment	Finds the tangent line to a point on the segment
SubPath	Finds the tangent line to a point on the segment

GetUserClick Method

Select one of the available subtopics below to see detailed help on **GetUserClick** method

Object	Method description
CorelScript	
Document	Gets a user click position and information about the state of the CTRL, ALT and SHIFT keys at the moment of click

Group Method

Select one of the available subtopics below to see detailed help on **Group** method

Object	Method description
CorelScript	
Shape	Creates a group; only works if the shape type is a Selection
ShapeRange	Creates a group

ImportWorkspace Method

Select one of the available subtopics below to see detailed help on **ImportWorkspace** method

Object	Method description
Application	Imports workspace elements into the current workspace
FrameWork	Imports a workspace into the framework

InsertPages Method

Select one of the available subtopics below to see detailed help on **InsertPages** method

Object	Method description
CorelScript	
Document	Adds a page or pages between two others; returns the first page added

Load Method

Select one of the available subtopics below to see detailed help on **Load** method

Object	Method description
PatternFill	Loads Pattern file
PDFVBASettings	
PrintSettings	method Load

Move Method

Select one of the available subtopics below to see detailed help on **Move** method

Object	Method description
FountainColor	Sets the Position
Node	Moves the node by a given delta
NodeRange	Moves all the nodes within the range by a given delta
Shape	Moves the shape using a delta
ShapeRange	Moves
SubPath	Moves the whole subpath by a given delta

Next Method

Select one of the available subtopics below to see detailed help on **Next** method

Object	Method description
Node	Gets the node following this one
Segment	Gets the segment following this one
SubPath	Gets the subpath following this one

Open Method

Select one of the available subtopics below to see detailed help on **Open** method

Object	Method description
Palettes	Open
RecentFile	Opens the file for editing

OrderBackOf Method

Select one of the available subtopics below to see detailed help on **OrderBackOf** method

Object	Method description
Shape	Arranges the shape's stacking order by moving it in back of another shape
ShapeRange	Arranges the shape's stacking order by moving it in back of another shape

OrderBackOne Method

Select one of the available subtopics below to see detailed help on **OrderBackOne** method

Object	Method description
CorelScript	
Shape	Arranges the shape's stacking order by moving it back one
ShapeRange	Arranges the shape's stacking order by moving it back one

OrderForwardOne Method

Select one of the available subtopics below to see detailed help on **OrderForwardOne** method

Object	Method description
CorelScript	
Shape	Arranges the shape's stacking order by moving it forward one
ShapeRange	Arranges the shape's stacking order by moving it forward one

OrderFrontOf Method

Select one of the available subtopics below to see detailed help on **OrderFrontOf** method

Object	Method description
Shape	Arranges the shape's stacking order by moving it in front of another shape
ShapeRange	Arranges the shape's stacking order by moving it in front of another shape

OrderReverse Method

Select one of the available subtopics below to see detailed help on **OrderReverse** method

Object	Method description
Shape	Reverses
ShapeRange	Reverses

OrderToBack Method

Select one of the available subtopics below to see detailed help on **OrderToBack** method

Object	Method description
CorelScript	
Shape	Arranges the shape's stacking order by moving it to the back
ShapeRange	Arranges the shape's stacking order by moving it to the back

OrderToFront Method

Select one of the available subtopics below to see detailed help on **OrderToFront** method

Object	Method description
CorelScript	
Shape	Arranges the shape's stacking order by moving it to the front
ShapeRange	Arranges the shape's stacking order by moving it to the front

Previous Method

Select one of the available subtopics below to see detailed help on **Previous** method

Object	Method description
Node	Gets the node preceding this one
Segment	Gets the segment preceding this one
SubPath	Gets the subpath preceding this one

PrintOut Method

Select one of the available subtopics below to see detailed help on **PrintOut** method

Object	Method description
Document	Opens the Print dialog
PrintJob	method PrintOut

Range Method

Select one of the available subtopics below to see detailed help on **Range** method

Object	Method description
Nodes	Creates a range containing the specified nodes
Segments	Creates a range containing the specified segments
Shapes	Gets the shape range from index

Redo Method

Select one of the available subtopics below to see detailed help on **Redo** method

Object	Method description
CorelScript	
Document	Reverses the last undo operation

Refresh Method

Select one of the available subtopics below to see detailed help on **Refresh** method

Object	Method description
Window	Refresh the window
Windows	Refresh all the windows

Remove Method

Select one of the available subtopics below to see detailed help on **Remove** method

Object	Method description
ArrowHeads	Removes an arrow head from the collection
NodeRange	Removes a node from the range
OutlineStyles	Removes an outline style
PatternCanvases	Removes a PatternCanvas from the PatternCanvases collection
SegmentRange	Removes a segment from the range
ShapeRange	Removes

RemoveAll Method

Select one of the available subtopics below to see detailed help on **RemoveAll** method

Object	Method description
NodeRange	Removes all nodes from the range
SegmentRange	Removes all segments from the range
ShapeRange	Removes All

RemoveFromContainer Method

Select one of the available subtopics below to see detailed help on **RemoveFromContainer** method

Object	Method description
Shape	Removes this shape from a power clip
ShapeRange	Removes this shape from a power clip

Repeat Method

Select one of the available subtopics below to see detailed help on **Repeat** method

Object	Method description
CorelScript	
Document	Repeats the last operation

Reset Method

Select one of the available subtopics below to see detailed help on **Reset** method

Object	Method description
CommandBar	Resets the control to it's default property values
PDFVBASettings	
PrintSettings	method Reset

ReverseDirection Method

Select one of the available subtopics below to see detailed help on **ReverseDirection** method

Object	Method description
Curve	Reverses the node order within the curve
SubPath	Reverses the node order within the subpath

Rotate Method

Select one of the available subtopics below to see detailed help on **Rotate** method

Object	Method description
EffectExtrude	Rotates the extrude
NodeRange	Rotates the nodes within the range as a shape
Shape	Rotates the shape by adding the amount to the current rotation value
ShapeRange	Rotates the shape by adding the amount to the current rotation value

RotateEx Method

Select one of the available subtopics below to see detailed help on **RotateEx** method

Object	Method description
Shape	Rotate Ex
ShapeRange	EX-Rotates the shape by adding the amount to the current rotation value

Save Method

Select one of the available subtopics below to see detailed help on **Save** method

Object	Method description
Document	Saves the document
OutlineStyles	Saves changes made to outline styles
Palette	Saves the palette
PDFVBASettings	
PrintSettings	method Save

Select Method

Select one of the available subtopics below to see detailed help on **Select** method

Object	Method description
EffectEnvelope	Selects an index for the envelope
PatternCanvas	Sets the image from preset image list
PostScriptFill	Selects the PostScript Fill by name or index
TextureFill	Selects the Library and the Texture

Selection Method

Select one of the available subtopics below to see detailed help on **Selection** method

Object	Method description
Curve	Creates a range containing all the selected nodes on this curve
Document	Gets a shape with a cdrSelectionShape type
SubPath	Creates a range containing all the select nodes on this subpath

Separate Method

Select one of the available subtopics below to see detailed help on **Separate** method

Object	Method description
CorelScript	
Effect	Separates the blend effect for the object
Shape	Separates combined shapes or a link group

SetBoundingBox Method

Select one of the available subtopics below to see detailed help on **SetBoundingBox** method

Object	Method description
Shape	Sets Bounding Box
ShapeRange	Sets Bounding Box

SetPosition Method

Select one of the available subtopics below to see detailed help on **SetPosition** method

Object	Method description
CorelScript	
Node	Sets the horizontal and vertical position of the node on the page
Shape	Sets Position
ShapeRange	Sets Position
SubPath	Sets the horizontal and vertical position of the subpath's head on the page

SetProperties Method

Select one of the available subtopics below to see detailed help on **SetProperties** method

Object	Method description
Outline	Sets all outline properties
PostScriptFill	Sets PostScript Fill Properties
TextureFill	Sets Texture Properties

SetRadius Method

Select one of the available subtopics below to see detailed help on **SetRadius** method

Object	Method description
Ellipse	Sets the ellipse radius
Rectangle	Sets the corners radius

SetRotationCenter Method

Select one of the available subtopics below to see detailed help on **SetRotationCenter** method

Object	Method description
Shape	Sets Rotation Center
ShapeRange	Sets the rotation center of the shape range

SetSize Method

Select one of the available subtopics below to see detailed help on **SetSize** method

Object	Method description
CorelScript	
Page	Sets the size of the page
Shape	Sets Size
ShapeRange	Sets Size

SetSizeEx Method

Select one of the available subtopics below to see detailed help on **SetSizeEx** method

Object	Method description
Shape	Sets the shape size using the anchor point
ShapeRange	Sets the shape range size using the anchor point

SetType Method

Select one of the available subtopics below to see detailed help on **SetType** method

Object	Method description
NodeRange	Sets the type of all nodes
SegmentRange	Sets the type of all segments

ShowDialog Method

Select one of the available subtopics below to see detailed help on **ShowDialog** method

Object	Method description
PDFVBASettings	
Printer	method ShowDialog
PrintSettings	method ShowDialog

Skew Method

Select one of the available subtopics below to see detailed help on **Skew** method

Object	Method description
NodeRange	Skew the nodes within the range as a shape
Shape	Skews the shape
ShapeRange	Skews the shape

SkewEx Method

Select one of the available subtopics below to see detailed help on **SkewEx** method

Object	Method description
Shape	Skew Ex
ShapeRange	EX-Skews the shape

Stretch Method

Select one of the available subtopics below to see detailed help on **Stretch** method

Object	Method description
NodeRange	Resizes the nodes within the range as a shape
Shape	Stretches the shape
ShapeRange	Stretches the shapes in the shape range

StretchEx Method

Select one of the available subtopics below to see detailed help on **StretchEx** method

Object	Method description
Shape	Stretches the shape using the anchor point
ShapeRange	Stretches the shapes in the shape range using the anchor point

Translate Method

Select one of the available subtopics below to see detailed help on **Translate** method

Object	Method description
CorelScriptFile	Translates a Corel SCRIPT
FountainFill	Moves Start and End points

Trim Method

Select one of the available subtopics below to see detailed help on **Trim** method

Object	Method description
CorelScript	
Shape	Trims to another object

Undo Method

Select one of the available subtopics below to see detailed help on **Undo** method

Object	Method description
CorelScript	
Document	Reverses the last operation

Unfreeze Method

Select one of the available subtopics below to see detailed help on **Unfreeze** method

Object	Method description
EffectLens	Unfreezes the lens effect
Transparency	Unfreezes the transparency

Ungroup Method

Select one of the available subtopics below to see detailed help on **Ungroup** method

Object	Method description
CorelScript	
EffectLens	Ungroups the lens object
Shape	Ungroups a group

UngroupAll Method

Select one of the available subtopics below to see detailed help on **UngroupAll** method

Object	Method description
CorelScript	
Shape	Ungroups all nested groups
ShapeRange	Ungroups all nested groups

Weld Method

Select one of the available subtopics below to see detailed help on **Weld** method

Object	Method description
CorelScript	
Shape	Welds to another object

AfterPrint Event

Select one of the available subtopics below to see detailed help on **AfterPrint** event

Object	Event description
Document	Event triggered after the Print dialog closes
GlobalDocument	Event triggered after the Print dialog closes

AfterSave Event

Select one of the available subtopics below to see detailed help on **AfterSave** event

Object	Event description
Document	Event triggered after the save is completed
GlobalDocument	Event triggered after the save is completed

AfterSaveAs Event

Select one of the available subtopics below to see detailed help on **AfterSaveAs** event

Object	Event description
Document	Event triggered after the save is completed
GlobalDocument	Event triggered after the save is completed

BeforeClose Event

Select one of the available subtopics below to see detailed help on **BeforeClose** event

Object	Event description
Document	Event triggered before the document closes
GlobalDocument	Event triggered before the document closes

BeforePrint Event

Select one of the available subtopics below to see detailed help on **BeforePrint** event

Object	Event description
Document	Event triggered before the Print dialog appears
GlobalDocument	Event triggered before the Print dialog appears

BeforeSave Event

Select one of the available subtopics below to see detailed help on **BeforeSave** event

Object	Event description
Document	Event triggered before the save dialog appears
GlobalDocument	Event triggered before the save dialog appears

BeforeSaveAs Event

Select one of the available subtopics below to see detailed help on **BeforeSaveAs** event

Object	Event description
Document	Event triggered before the save dialog appears
GlobalDocument	Event triggered before the save dialog appears

LayerCreated Event

Select one of the available subtopics below to see detailed help on **LayerCreated** event

Object	Event description
Document	Event triggered after layer creation
GlobalDocument	Event triggered after layer creation

LayerDeleted Event

Select one of the available subtopics below to see detailed help on **LayerDeleted** event

Object	Event description
Document	Event triggered after layer deletion
GlobalDocument	Event triggered after layer deletion

LayerSelected Event

Select one of the available subtopics below to see detailed help on **LayerSelected** event

Object	Event description
Document	Event triggered after layer selection
GlobalDocument	Event triggered after layer selection

PageCreated Event

Select one of the available subtopics below to see detailed help on **PageCreated** event

Object	Event description
Document	Event triggered after page creation
GlobalDocument	Event triggered after page creation

PageDeleted Event

Select one of the available subtopics below to see detailed help on **PageDeleted** event

Object	Event description
Document	Event triggered after page deletion
GlobalDocument	Event triggered after page deletion

PageSelected Event

Select one of the available subtopics below to see detailed help on **PageSelected** event

Object	Event description
Document	Event triggered after page selection
GlobalDocument	Event triggered after page selection

SelectionChanged Event

Select one of the available subtopics below to see detailed help on **SelectionChanged** event

Object	Event description
Document	Event triggered after selection changed
GlobalDocument	Event triggered after selection changed

ShapeCreated Event

Select one of the available subtopics below to see detailed help on **ShapeCreated** event

Object	Event description
AddinHook	
Document	Event triggered after shape creation
GlobalDocument	Event triggered after shape creation

ShapeDeleted Event

Select one of the available subtopics below to see detailed help on **ShapeDeleted** event

Object	Event description
Document	Event triggered after shape deletion
GlobalDocument	Event triggered after shape deletion

ShapeMoved Event

Select one of the available subtopics below to see detailed help on **ShapeMoved** event

Object	Event description
Document	Event triggered after shape creation
GlobalDocument	Event triggered after shape creation

cdrAddinFilter

Enum **cdrAddinFilter**

Referenced By

Constant	Value	Description
cdrAddinFilterNone	0	
cdrAddinFilterWhileDrawing	1	
cdrAddinFilterShapeCreated	2	
cdrAddinFilterNew	4	
cdrAddinFilterExecute	8	

{button ,AL(`CLS_cdrAddinFilter')} **Related Topics**

cdrAlignment

Enum **cdrAlignment**

Referenced By

Constant	Value	Description
cdrNoAlignment	0	
cdrLeftAlignment	1	
cdrRightAlignment	2	
cdrCenterAlignment	3	
cdrFullJustifyAlignment	4	
cdrForceJustifyAlignment	5	
cdrMixedAlignment	6	

{button ,AL(`CLS_cdrAlignment')} **Related Topics**

cdrAntiAliasingType

Enum **cdrAntiAliasingType**

Referenced By

Constant	Value	Description
cdrNoAntiAliasing	0	
cdrNormalAntiAliasing	1	
cdrSupersampling	2	

{button ,AL(`CLS_cdrAntiAliasingType')} **Related Topics**

cdrBlendMode

Enum **cdrBlendMode**

Referenced By

Constant	Value	Description
cdrBlendSteps	0	
cdrBlendSpacing	1	

{button ,AL(`CLS_cdrBlendMode')} **Related Topics**

cdrColorType

Enum **cdrColorType**

Referenced By

Constant	Value	Description
cdrColorPantone	1	
cdrColorCMYK	2	
cdrColorCMY	4	
cdrColorRGB	5	
cdrColorHSB	6	
cdrColorHLS	7	
cdrColorBlackAndWhite	8	
cdrColorGray	9	
cdrColorYIQ	11	
cdrColorLab	12	
cdrColorPantoneHex	14	
cdrColorRegistration	20	
cdrColorSpot	25	
cdrColorMixed	99	

{button ,AL(`CLS_cdrColorType')} **Related Topics**

cdrCompressionType

Enum **cdrCompressionType**

Referenced By

Constant	Value	Description
cdrCompressionNone	0	
cdrCompressionLZW	1	
cdrCompressionPackBits	2	
cdrCompressionHuffman	3	
cdrCompressionCCITT3_1d	4	
cdrCompressionCCITT3_2d	5	
cdrCompressionCCITT4	6	
cdrCompressionRLE_LW	7	
cdrCompressionJPEG	8	

{button ,AL(`CLS_cdrCompressionType')} **Related Topics**

cdrContourDirection

Enum **cdrContourDirection**

Referenced By

Constant	Value	Description
cdrContourInside	0	
cdrContourOutside	1	
cdrContourToCenter	2	

{button ,AL(`CLS_cdrContourDirection')} **Related Topics**

cdrCursorShape

Enum **cdrCursorShape**

Referenced By

Constant	Value	Description
cdrCursorWinAppStarting	0	
cdrCursorWinArrow	1	
cdrCursorWinCross	2	
cdrCursorWinHelp	3	
cdrCursorWinIbeam	4	
cdrCursorWinNo	5	
cdrCursorWinSizeAll	6	
cdrCursorWinSizeNeSw	7	
cdrCursorWinSizeNs	8	
cdrCursorWinSizeNwSe	9	
cdrCursorWinSizeWe	10	
cdrCursorWinUpArrow	11	
cdrCursorWinWait	12	
cdrCursorSmallcrosshair	1	
cdrCursorPick	5	
cdrCursorNodeEdit	6	
cdrCursorEyeDrop	34	
cdrCursorExtPick	60	
cdrCursorPickNone	79	
cdrCursorPenJoin	91	
cdrCursorPickOvertarget	107	
cdrCursorTrimSingle	139	
cdrCursorWeldSingle	141	
cdrCursorIntersectSingle	143	

{button ,AL(`CLS_cdrCursorShape')} **Related Topics**

cdrDataFormatType

Enum **cdrDataFormatType**

Referenced By

Constant	Value	Description
cdrFormatGeneral	0	
cdrFormatDataTime	1	
cdrFormatLinearAngular	2	
cdrFormatNumeric	3	

{button ,AL(`CLS_cdrDataFormatType')} **Related Topics**

cdrDistortionType

Enum **cdrDistortionType**

Referenced By

Constant	Value	Description
cdrDistortionPushPull	0	
cdrDistortionZipper	1	
cdrDistortionTwister	2	

{button ,AL(`CLS_cdrDistortionType')} **Related Topics**

cdrDitherType

Enum **cdrDitherType**

Referenced By

Constant	Value	Description
cdrDitherNone	0	
cdrDitherOrdered	1	
cdrDitherJarvis	2	
cdrDitherStucki	3	
cdrDitherFloyd	4	

{button ,AL(`CLS_cdrDitherType')} **Related Topics**

cdrDropShadowType

Enum **cdrDropShadowType**

Referenced By

Constant	Value	Description
cdrDropShadowFlat	0	
cdrDropShadowBottom	1	
cdrDropShadowTop	2	
cdrDropShadowLeft	3	
cdrDropShadowRight	4	

{button ,AL(`CLS_cdrDropShadowType')} **Related Topics**

cdrEdgeType

Enum **cdrEdgeType**

Referenced By

Constant	Value	Description
cdrEdgeLinear	0	
cdrEdgeSquared	1	
cdrEdgeFlat	2	
cdrEdgeInverseSquared	3	
cdrEdgeMesa	4	
cdrEdgeGaussian	5	

{button ,AL(`CLS_cdrEdgeType')} **Related Topics**

cdrEffectType

Enum **cdrEffectType**

Referenced By

Constant	Value	Description
cdrBlend	0	
cdrExtrude	1	
cdrEnvelope	2	
cdrTextOnPath	3	
cdrControlPath	4	
cdrDropShadow	5	
cdrContour	6	
cdrDistortion	7	
cdrPerspective	8	
cdrLens	9	

{button ,AL(`CLS_cdrEffectType')} **Related Topics**

cdrEllipseType

Enum **cdrEllipseType**

Referenced By

Constant	Value	Description
cdrEllipse	0	
cdrArc	1	
cdrPie	2	

{button ,AL(`CLS_cdrEllipseType')} **Related Topics**

cdrEnvelopeMode

Enum **cdrEnvelopeMode**

Referenced By

Constant	Value	Description
cdrEnvelopeHorizontal	0	
cdrEnvelopeOriginal	1	
cdrEnvelopePutty	2	
cdrEnvelopeVertical	3	

{button ,AL(`CLS_cdrEnvelopeMode')} **Related Topics**

cdrExportRange

Enum **cdrExportRange**

Referenced By

Constant	Value	Description
cdrAllPages	0	
cdrCurrentPage	1	
cdrSelection	2	

{button ,AL(`CLS_cdrExportRange')} **Related Topics**

cdrExtrudeLightPosition

Enum **cdrExtrudeLightPosition**

Referenced By

Constant	Value	Description
cdrLightFrontTopLeft	0	
cdrLightFrontTop	1	
cdrLightFrontTopRight	2	
cdrLightFrontLeft	3	
cdrLightFrontCenter	4	
cdrLightFrontRight	5	
cdrLightFrontBottomLeft	6	
cdrLightFrontBottom	7	
cdrLightFrontBottomRight	8	
cdrLightBackTopLeft	9	
cdrLightBackTop	10	
cdrLightBackTopRight	11	
cdrLightBackRight	14	
cdrLightBackBottomRight	17	

{button ,AL(`CLS_cdrExtrudeLightPosition')} **Related Topics**

cdrExtrudeShading

Enum **cdrExtrudeShading**

Referenced By

Constant	Value	Description
cdrExtrudeObjectFill	0	
cdrExtrudeSolidFill	1	
cdrExtrudeColorShading	2	

{button ,AL(`CLS_cdrExtrudeShading')} **Related Topics**

cdrExtrudeType

Enum **cdrExtrudeType**

Referenced By

Constant	Value	Description
cdrExtrudeSmallBack	0	
cdrExtrudeSmallFront	1	
cdrExtrudeBigBack	2	
cdrExtrudeBigFront	3	
cdrExtrudeBackParallel	4	
cdrExtrudeFrontParallel	5	

{button ,AL(`CLS_cdrExtrudeType')} **Related Topics**

cdrExtrudeVPType

Enum **cdrExtrudeVPType**

Referenced By

Constant	Value	Description
cdrVPLockedToShape	0	
cdrVPLockedToPage	1	
cdrVPShared	2	

{button ,AL(`CLS_cdrExtrudeVPType')} **Related Topics**

cdrFeatherType

Enum **cdrFeatherType**

Referenced By

Constant	Value	Description
cdrFeatherInside	0	
cdrFeatherMiddle	1	
cdrFeatherOutside	2	
cdrFeatherAverage	3	

{button ,AL(`CLS_cdrFeatherType')} **Related Topics**

cdrFileVersion

Enum **cdrFileVersion**

Referenced By

Constant	Value	Description
cdrCurrentVersion	0	
cdrVersion5	5	
cdrVersion6	6	
cdrVersion7	7	
cdrVersion8	8	
cdrVersion9	9	
cdrVersion10	10	

{button ,AL(`CLS_cdrFileVersion')} **Related Topics**

cdrFillType

Enum **cdrFillType**

Referenced By

Constant	Value	Description
cdrNoFill	0	
cdrUniformFill	1	
cdrFountainFill	2	
cdrPostscriptFill	3	
cdrMonoBitmapFill	4	
cdrReservedFill	5	
cdrColorBitmapFill	6	
cdrVectorFill	7	
cdrTextureFill	8	
cdrPatternFill	9	

{button ,AL(`CLS_cdrFillType')} **Related Topics**

cdrFilter

Enum **cdrFilter**

Referenced By

Constant	Value	Description
cdrAutoSense	0	
cdr3DMF	1559	3D Model
cdrAI	1305	Adobe Illustrator
cdrAT1	1303	Adobe Type 1 font
cdrAVI	1536	Video for Windows
cdrBarista	1312	Corel Barista
cdrBMP	769	Windows Bitmap
cdrCAL	800	CALS Compressed Bitmap
cdrCCD	1798	CorelCAD Thumbnail
cdrCCH	1541	CorelCHART
cdrCDR	1795	CorelDRAW Document
cdrCDT	1800	CorelDRAW Template
cdrCDX	1796	CorelDRAW Compressed
cdrCFL	1550	
cdrCGM	1280	Computer Graphics Metafile
cdrCLK	1802	Corel R.A.V.E.
cdrCMF	1295	Corel Metafile
cdrCMV	1539	Corel MOVE
cdrCMX5	1794	Corel CMX 5
cdrCMX6	1793	Corel CMX 6
cdrCPH	1803	Corel PrintHouse
cdrCPT	1792	Core PHOTO-PAINT
cdrCPT7	1799	Core PHOTO-PAINT 7
cdrCPT9	1808	Core PHOTO-PAINT 9
cdrCPT10	1808	Core PHOTO-PAINT 10
cdrCPX	1797	Corel CMX Compressed
cdrCUR	785	Cursor File
cdrDOC	2068	Microsoft Word 97/2000
cdrDRW	1282	Micrografx 2/3
cdrDWG	1328	AutoCAD
cdrDXF	1296	AutoCAD
cdrEMF	1300	Enhanced Metafile
cdrEPS	1289	Encapsulated PostScript
cdrEPSPhotoPaint	804	Encapsulated PostScript (Corel PHOTO-PAINT)
cdrEXE	786	Bitmap Resources in EXE and DLL
cdrFH	1344	Macromedia FreeHand

cdrFMV	1329	Frame Vector Metafile
cdrFPX	806	Kodak FlashPix Image
cdrGEM	1284	GEM Paint
cdrGIF	773	CompuServe Graphics
cdrGIFAnimation	1558	Animated CompuServe Graphics
cdrHPGL	1281	HPGL Plotter File
cdrICO	784	Icon File
cdrIMG	787	GEM Paint
cdrJPEG	774	JPEG
cdrLotus123	2066	Lotus 1-2-3
cdrMAC	791	MACPaint Bitmap
cdrMacWord5	2052	Microsoft Word for Macintosh 5
cdrMET	1291	MET Metafile
cdrMPEG	1551	MPEG Movie
cdrNAP	1292	NAP Metafile
cdrOS2BMP	792	OS/2 Bitmap
cdrOS2Metafile	1291	OS/2 Metafile
cdrPAT	1801	CorelDRAW Pattern
cdrPCD	775	Kodak Photo-CD Image
cdrPCX	770	PaintBrush
cdrPDFPlaceable	1314	Placeable PDF
cdrPIC	790	Lotus Pic
cdrPICT	1288	Macintosh PICT
cdrPICT4	808	Macintosh PICT 4
cdrPICTWithEPS	1316	Encapsulated PostScript with PICT header
cdrPIF	1285	IBM PIF
cdrPLT	1281	Plotter File
cdrPNG	802	Portable Network Graphics file
cdrPNM	817	Portable Anymap file
cdrPP	789	Picture Publisher 4
cdrPP4	789	Picture Publisher 4
cdrPP5	803	Picture Publisher 5
cdrPSD	788	Adobe Photoshop
cdrPSEncapsulated	1289	Encapsulated PostScript
cdrPSInterpreted	1290	Interpreted PostScript
cdrQTM	1542	QuickTime movie
cdrQTVR	1560	QuickTime VR
cdrRAWPhotoPaint	805	Corel PHOTO-PAINT Raw file
cdrRIFF	807	Rich Image File Format
cdrRND	1297	
cdrRTF	2053	Rich Text Format
cdrSAM	2063	AmiPro

cdrSCD	1298	
cdrSCT	776	
cdrSHW	1540	
cdrSVG	1345	Scalable Vector Graphics
cdrSWF	1343	Macromedia Flash
cdrTGA	771	Targa Bitmap
cdrTIFF	772	Tagged Image File Format
cdrTTF	1302	TrueType Font
cdrTXT	2048	Text file
cdrVSD	1315	Visio Graphics
cdrWI	793	Wavelet Compressed Bitmap
cdrWK	2066	Lotus 1-2-3
cdrWMF	1294	Windows Metafile
cdrWord2	2050	Microsoft Word 2
cdrWord2000	2068	Microsoft Word 2000
cdrWord55	2051	Microsoft Word 5.5
cdrWord95	2049	Microsoft Word 95
cdrWordStar2000	2061	WordStar 2000
cdrWordStar7	2060	WordStar 7
cdrWP4	2058	WordPerfect 4
cdrWP50	2057	WordPerfect 5.0
cdrWP51	2056	WordPerfect 5.1
cdrWP9	2055	WordPerfect 9
cdrWPD	2055	WordPerfect Document
cdrWPG	1287	WordPerfect Graphics
cdrWPG2	1304	
cdrWPM	2072	Corel WordPerfect 1/2 for Macintosh
cdrWSD	2061	WordStar for Windows 1/2
cdrWSW	2059	WordStar for Windows 1/2
cdrWVL	793	Wavelet Compressed Bitmap
cdrXCF	816	Gimp Image
cdrXLS	2065	Microsoft Excel
cdrXPM	809	XPixMap Image
cdrXY	2062	XYWrite for Windows 4

{button ,AL(`CLS_cdrFilter')} **Related Topics**

cdrFittedOrientation

Enum **cdrFittedOrientation**

Referenced By

Constant	Value	Description
cdrRotateOrientation	0	
cdrVerticalSkewOrientation	1	
cdrHorizontalSkewOrientation	2	
cdrUprightOrientation	3	

{button ,AL(`CLS_cdrFittedOrientation')} **Related Topics**

cdrFittedPlacement

Enum **cdrFittedPlacement**

Referenced By

Constant	Value	Description
cdrLeftPlacement	0	
cdrRightPlacement	1	
cdrCenterPlacement	2	

{button ,AL(`CLS_cdrFittedPlacement')} **Related Topics**

cdrFittedQuadrant

Enum **cdrFittedQuadrant**

Referenced By

Constant	Value	Description
cdrLeftQuadrant	0	
cdrRightQuadrant	1	
cdrTopQuadrant	2	
cdrBottomQuadrant	3	

{button ,AL(`CLS_cdrFittedQuadrant')} **Related Topics**

cdrFittedVertPlacement

Enum **cdrFittedVertPlacement**

Referenced By

Constant	Value	Description
cdrCustomVertPlacement	0	
cdrBaselineVertPlacement	1	
cdrAscenderVertPlacement	2	
cdrDescenderVertPlacement	3	
cdrCenterVertPlacement	4	

{button ,AL(`CLS_cdrFittedVertPlacement')} **Related Topics**

cdrFlipAxes

Enum **cdrFlipAxes**

Referenced By

Constant	Value	Description
cdrFlipHorizontal	1	
cdrFlipVertical	2	
cdrFlipBoth	3	

{button ,AL(`CLS_cdrFlipAxes')} **Related Topics**

cdrFontCase

Enum **cdrFontCase**

Referenced By

Constant	Value	Description
cdrNormalFontCase	0	
cdrSmallCapsFontCase	1	
cdrAllCapsFontCase	2	
cdrMixedFontCase	3	

{button ,AL(`CLS_cdrFontCase')} **Related Topics**

cdrFontLine

Enum **cdrFontLine**

Referenced By

Constant	Value	Description
cdrNoFontLine	0	
cdrSingleThinFontLine	1	
cdrSingleThinWordFontLine	2	
cdrSingleThickFontLine	3	
cdrSingleThickWordFontLine	4	
cdrDoubleThinFontLine	5	
cdrDoubleThinWordFontLine	6	
cdrMixedFontLine	7	

{button ,AL(`CLS_cdrFontLine')} **Related Topics**

cdrFontPosition

Enum **cdrFontPosition**

Referenced By

Constant	Value	Description
cdrNormalFontPosition	0	
cdrSubscriptFontPosition	1	
cdrSuperscriptFontPosition	2	
cdrMixedFontPosition	3	

{button ,AL(`CLS_cdrFontPosition')} **Related Topics**

cdrFontStyle

Enum **cdrFontStyle**

Referenced By

Constant	Value	Description
cdrNormalFontStyle	0	
cdrBoldFontStyle	1	
cdrItalicFontStyle	2	
cdrBoldItalicFontStyle	3	
cdrThinFontStyle	4	
cdrThinItalicFontStyle	5	
cdrExtraLightFontStyle	6	
cdrExtraLightItalicFontStyle	7	
cdrMediumFontStyle	8	
cdrMediumItalicFontStyle	9	
cdrSemiBoldFontStyle	10	
cdrSemiBoldItalicFontStyle	11	
cdrExtraBoldFontStyle	12	
cdrExtraBoldItalicFontStyle	13	
cdrHeavyFontStyle	14	
cdrHeavyItalicFontStyle	15	
cdrMixedFontStyle	16	

{button ,AL(`CLS_cdrFontStyle')} **Related Topics**

cdrFountainFillBlendType

Enum **cdrFountainFillBlendType**

Referenced By

Constant	Value	Description
cdrDirectFountainFillBlend	0	
cdrRainbowCWFountainFillBlend	1	
cdrRainbowCCWFountainFillBlend	2	
cdrCustomFountainFillBlend	3	

{button ,AL(`CLS_cdrFountainFillBlendType')} **Related Topics**

cdrFountainFillType

Enum **cdrFountainFillType**

Referenced By

Constant	Value	Description
cdrLinearFountainFill	1	
cdrRadialFountainFill	2	
cdrConicalFountainFill	3	
cdrSquareFountainFill	4	

{button ,AL(`CLS_cdrFountainFillType')} **Related Topics**

cdrGridType

Enum **cdrGridType**

Referenced By

Constant	Value	Description
cdrGridDot	0	
cdrGridLine	1	

{button ,AL(`CLS_cdrGridType')} **Related Topics**

cdrGuideType

Enum **cdrGuideType**

Referenced By

Constant	Value	Description
cdrAllGuides	-1	
cdrHorizontalGuide	0	
cdrVerticalGuide	1	
cdrSlantedGuide	2	

{button ,AL(`CLS_cdrGuideType')} **Related Topics**

cdrImagePaletteType

Enum **cdrImagePaletteType**

Referenced By

Constant	Value	Description
cdrPaletteUniform	0	
cdrPaletteStdVGA	1	
cdrPaletteAdaptive	2	
cdrPaletteOptimized	3	
cdrPaletteBlackBody	4	
cdrPaletteGrayscale	5	
cdrPaletteSystem	6	
cdrPaletteIE	7	
cdrPaletteNetscape	8	
cdrPaletteCustom	9	

{button ,AL(`CLS_cdrImagePaletteType')} **Related Topics**

cdrImageType

Enum **cdrImageType**

Referenced By

Constant	Value	Description
cdrBlackAndWhiteImage	0	
cdr16ColorsImage	1	
cdrGrayscaleImage	2	
cdrPalettedImage	3	
cdrRGBColorImage	4	
cdrCMYKColorImage	5	
cdrDuotoneImage	6	

{button ,AL(`CLS_cdrImageType')} **Related Topics**

cdrLensType

Enum **cdrLensType**

Referenced By

Constant	Value	Description
cdrLensMagnify	0	
cdrLensFishEye	1	
cdrLensWireframe	2	
cdrLensColorLimit	3	
cdrLensColorAdd	4	
cdrLensInvert	5	
cdrLensBrighten	6	
cdrLensTintedGrayscale	7	
cdrLensHeatMap	8	
cdrLensTransparency	9	
cdrLensCustomColorMap	10	

{button ,AL(`CLS_cdrLensType')} **Related Topics**

cdrLineSpacingType

Enum **cdrLineSpacingType**

Referenced By

Constant	Value	Description
cdrPercentOfCharacterHeightLineSpacing	0	
cdrPointLineSpacing	1	
cdrPercentOfPointSizeLineSpacing	2	
cdrMixedLineSpacing	3	

{button ,AL(`CLS_cdrLineSpacingType')} **Related Topics**

cdrNodeType

Enum **cdrNodeType**

Referenced By

Constant	Value	Description
cdrCuspNode	0	
cdrSmoothNode	1	
cdrSymmetricalNode	2	
cdrMixedNodes	3	

{button ,AL(`CLS_cdrNodeType')} **Related Topics**

cdrOptimization

Enum **cdrOptimization**

Referenced By

Constant	Value	Description
cdrOptimizationOff	0	
cdrOptimizationOn	1	

{button ,AL(`CLS_cdrOptimization')} **Related Topics**

cdrOutlineLineCaps

Enum **cdrOutlineLineCaps**

Referenced By

Constant	Value	Description
cdrOutlineButtLineCaps	0	
cdrOutlineRoundLineCaps	1	
cdrOutlineSquareLineCaps	2	

{button ,AL(`CLS_cdrOutlineLineCaps')} **Related Topics**

cdrOutlineLineJoin

Enum **cdrOutlineLineJoin**

Referenced By

Constant	Value	Description
cdrOutlineMiterLineJoin	0	
cdrOutlineRoundLineJoin	1	
cdrOutlineBevelLineJoin	2	

{button ,AL(`CLS_cdrOutlineLineJoin')} **Related Topics**

cdrOutlineType

Enum **cdrOutlineType**

Referenced By

Constant	Value	Description
cdrNoOutline	0	
cdrOutline	1	

{button ,AL(`CLS_cdrOutlineType')} **Related Topics**

cdrPageBackground

Enum **cdrPageBackground**

Referenced By

Constant	Value	Description
cdrPageBackgroundNone	0	
cdrPageBackgroundSolid	1	
cdrPageBackgroundBitmap	2	

{button ,AL(`CLS_cdrPageBackground')} **Related Topics**

cdrPageOrientation

Enum **cdrPageOrientation**

Referenced By

Constant	Value	Description
cdrPortrait	0	
cdrLandscape	1	

{button ,AL(`CLS_cdrPageOrientation')} **Related Topics**

cdrPaletteID

Enum **cdrPaletteID**

Referenced By

Constant	Value	Description
cdrTRUMATCH	1	
cdrPANTONEProcess	2	
cdrPANTONECorel8	3	
cdrUniform	7	
cdrFOCOLTONE	8	
cdrSpectraMaster	9	
cdrTOYO	10	
cdrDIC	11	
cdrPANTONEHexCoated	12	
cdrPANTONEHexUncoated	24	
cdrLab	13	
cdrNetscapeNavigator	14	
cdrInternetExplorer	15	
cdrPANTONECoated	17	
cdrPANTONEUncoated	18	
cdrPANTONEMetallic	20	
cdrPANTONEPastelCoated	21	
cdrPANTONEPastelUncoated	22	
cdrHKS	23	
cdrCustom	0	

{button ,AL(`CLS_cdrPaletteID')} **Related Topics**

cdrPaletteType

Enum **cdrPaletteType**

Referenced By

Constant	Value	Description
cdrFixedPalette	0	
cdrCustomPalette	1	

{button ,AL(`CLS_cdrPaletteType')} **Related Topics**

cdrPanoseMatchingType

Enum **cdrPanoseMatchingType**

Referenced By

Constant	Value	Description
cdrPanosePrompt	0	
cdrPanoseTemporary	1	
cdrPanosePermanent	2	

{button ,AL(`CLS_cdrPanoseMatchingType')} **Related Topics**

cdrPatternCanvasSize

Enum **cdrPatternCanvasSize**

Referenced By

Constant	Value	Description
cdrPatternCanvas16x16	0	
cdrPatternCanvas32x32	1	
cdrPatternCanvas64x64	2	
cdrPatternCanvasCustom	3	

{button ,AL(`CLS_cdrPatternCanvasSize')} **Related Topics**

cdrPatternFillType

Enum **cdrPatternFillType**

Referenced By

Constant	Value	Description
cdrTwoColorPattern	0	
cdrFullColorPattern	1	
cdrBitmapPattern	2	

{button ,AL(`CLS_cdrPatternFillType')} **Related Topics**

cdrPointType

Enum **cdrPointType**

Referenced By

Constant	Value	Description
cdrShapePoint	0	
cdrFreePoint	1	

{button ,AL(`CLS_cdrPointType')} **Related Topics**

cdrPolygonType

Enum **cdrPolygonType**

Referenced By

Constant	Value	Description
cdrPolygon	0	
cdrStar	1	

{button ,AL(`CLS_cdrPolygonType')} **Related Topics**

cdrPositionOfPointOverShape

Enum **cdrPositionOfPointOverShape**

Referenced By

Constant	Value	Description
cdrOutsideShape	0	
cdrOnMarginOfShape	1	
cdrInsideShape	2	

{button ,AL(`CLS_cdrPositionOfPointOverShape')} **Related Topics**

cdrPresetPoint

Enum **cdrPresetPoint**

Referenced By

Constant	Value	Description
cdrTopLeftPoint	-1	
cdrTopPoint	-2	
cdrTopRightPoint	-3	
cdrRightPoint	-4	
cdrBottomRightPoint	-5	
cdrBottomPoint	-6	
cdrBottomLeftPoint	-7	
cdrLeftPoint	-8	
cdrCenterPoint	-9	
cdrFirstPoint	-10	
cdrLastPoint	-11	

{button ,AL(`CLS_cdrPresetPoint')} **Related Topics**

cdrReferencePoint

Enum **cdrReferencePoint**

Referenced By

Constant	Value	Description
cdrTopRight	1	
cdrTopMiddle	2	
cdrTopLeft	3	
cdrMiddleLeft	4	
cdrBottomLeft	5	
cdrBottomMiddle	6	
cdrBottomRight	7	
cdrMiddleRight	8	
cdrCenter	9	

{button ,AL(`CLS_cdrReferencePoint')} **Related Topics**

cdrSegmentOffsetType

Enum **cdrSegmentOffsetType**

Referenced By

Constant	Value	Description
cdrAbsoluteSegmentOffset	0	
cdrRelativeSegmentOffset	1	
cdrParamSegmentOffset	2	

{button ,AL(`CLS_cdrSegmentOffsetType')} **Related Topics**

cdrSegmentType

Enum **cdrSegmentType**

Referenced By

Constant	Value	Description
cdrLineSegment	0	
cdrCurveSegment	1	
cdrMixedSegments	2	

{button ,AL(`CLS_cdrSegmentType')} **Related Topics**

cdrShapeEnumDirection

Enum **cdrShapeEnumDirection**

Referenced By

Constant	Value	Description
cdrShapeEnumTopFirst	0	
cdrShapeEnumBottomFirst	1	

{button ,AL(`CLS_cdrShapeEnumDirection')} **Related Topics**

cdrShapeLevel

Enum **cdrShapeLevel**

Referenced By

Constant	Value	Description
cdrLevelGroup	0	
cdrLevelContainer	1	
cdrLevelLayer	2	
cdrLevelPage	3	
cdrLevelDocument	4	

{button ,AL(`CLS_cdrShapeLevel')} **Related Topics**

cdrShapeType

Enum **cdrShapeType**

Referenced By

Constant	Value	Description
cdrNoShape	0	
cdrRectangleShape	1	
cdrEllipseShape	2	
cdrCurveShape	3	
cdrPolygonShape	4	
cdrBitmapShape	5	
cdrTextShape	6	
cdrGroupShape	7	
cdrSelectionShape	8	
cdrGuidelineShape	9	
cdrBlendGroupShape	10	
cdrExtrudeGroupShape	11	
cdrOLEObjectShape	12	
cdrContourGroupShape	13	
cdrLinearDimensionShape	14	
cdrBevelGroupShape	15	
cdrDropShadowGroupShape	16	
cdr3DObjectShape	17	
cdrArtisticMediaGroupShape	18	
cdrConnectorShape	19	
cdrMeshFillShape	20	

{button ,AL(`CLS_cdrShapeType')} **Related Topics**

cdrSpiralType

Enum **cdrSpiralType**

Referenced By

Constant	Value	Description
cdrSymmetric	0	
cdrLogarithmic	1	

{button ,AL(`CLS_cdrSpiralType')} **Related Topics**

cdrTextFrames

Enum **cdrTextFrames**

Referenced By

Constant	Value	Description
cdrThisFrameOnly	0	
cdrStartAtThisFrame	1	
cdrAllFrames	2	

{button ,AL(`CLS_cdrTextFrames')} **Related Topics**

cdrTextIndexingType

Enum **cdrTextIndexingType**

Referenced By

Constant	Value	Description
cdrCharacterIndexing	0	
cdrWordIndexing	1	
cdrParagraphIndexing	2	

{button ,AL(`CLS_cdrTextIndexingType')} **Related Topics**

cdrTextType

Enum **cdrTextType**

Referenced By

Constant	Value	Description
cdrArtisticText	0	
cdrParagraphText	1	
cdrArtisticFittedText	2	
cdrParagraphFittedText	3	

{button ,AL(`CLS_cdrTextType')} **Related Topics**

cdrTexturePropertyType

Enum **cdrTexturePropertyType**

Referenced By

Constant	Value	Description
cdrTexturePropertyNumeric	0	
cdrTexturePropertyColorRGB	1	
cdrTexturePropertyColorHSB	2	
cdrTexturePropertyColorCMYK	3	

{button ,AL(`CLS_cdrTexturePropertyType')} **Related Topics**

cdrThumbnailSize

Enum **cdrThumbnailSize**

Referenced By

Constant	Value	Description
cdrNoThumbnail	0	
cdr1KMonoThumbnail	1	
cdr5KColorThumbnail	2	
cdr10KColorThumbnail	3	

{button ,AL(`CLS_cdrThumbnailSize')} **Related Topics**

cdrTileOffsetType

Enum **cdrTileOffsetType**

Referenced By

Constant	Value	Description
cdrTileOffsetRow	0	
cdrTileOffsetColumn	1	

{button ,AL(`CLS_cdrTileOffsetType')} **Related Topics**

cdrTools

Enum **cdrTools**

Referenced By

Constant	Value	Description
cdrToolNone	0	
cdrToolPick	1	
cdrToolNodeEdit	2	
cdrToolKnife	64	
cdrToolBezierKnife	81	
cdrToolEraser	68	
cdrToolDrawRectangle	3	
cdrToolDrawEllipse	4	
cdrToolDrawText	7	
cdrToolDrawFreehand	5	
cdrToolDrawNaturalPen	103	
cdrToolDrawBezier	6	
cdrToolHorizontalDimension	54	
cdrToolVerticalDimension	55	
cdrToolAutoDimension	110	
cdrToolSlantedDimension	56	
cdrToolLeaderText	57	
cdrToolAngledDimension	58	
cdrToolConnectorLines	59	
cdrToolDrawPolygon	60	
cdrToolDrawSpiral	62	
cdrToolDrawGrid	63	
cdrToolZoom	66	
cdrToolPan	67	
cdrToolFill	71	
cdrToolTransparency	72	
cdrToolInteractiveExtrude	73	
cdrToolBlend	74	
cdrToolRotate	75	
cdrToolReflect	76	
cdrToolScale	77	
cdrToolSkew	78	
cdrToolDistortion	79	
cdrToolContour	113	
cdrToolInsertHTMLFormObject	111	
cdrToolDropShadow	80	

{button ,AL(`CLS_cdrTools')} **Related Topics**

cdrTransparencyAppliedTo

Enum **cdrTransparencyAppliedTo**

Referenced By

Constant	Value	Description
cdrApplyToFill	0	
cdrApplyToOutline	1	
cdrApplyToFillAndOutline	2	

{button ,AL(`CLS_cdrTransparencyAppliedTo')} **Related Topics**

cdrTransparencyType

Enum **cdrTransparencyType**

Referenced By

Constant	Value	Description
cdrNoTransparency	0	
cdrUniformTransparency	1	
cdrFountainTransparency	2	
cdrPatternTransparency	3	
cdrTextureTransparency	4	

{button ,AL(`CLS_cdrTransparencyType')} **Related Topics**

cdrUnit

Enum **cdrUnit**

Referenced By

Constant	Value	Description
cdrTenthMicron	0	
cdrInch	1	
cdrFoot	2	
cdrMillimeter	3	
cdrCentimeter	4	
cdrPixel	5	
cdrMile	6	
cdrMeter	7	
cdrKilometer	8	
cdrDidots	9	
cdrAgate	10	
cdrYard	11	
cdrPica	12	
cdrCicero	13	
cdrPoint	14	
cdrUnitQ	15	
cdrUnitH	16	

{button ,AL(`CLS_cdrUnit')} **Related Topics**

cdrURLRegion

Enum **cdrURLRegion**

Referenced By

Constant	Value	Description
cdrURLRegionDefault	0	
cdrURLRegionRectangle	1	
cdrURLRegionShape	2	

{button ,AL(`CLS_cdrURLRegion')} **Related Topics**

cdrViewType

Enum **cdrViewType**

Referenced By

Constant	Value	Description
cdrSimpleWireframeView	0	
cdrWireframeView	1	
cdrDraftView	2	
cdrNormalView	3	
cdrEnhancedView	4	

{button ,AL(`CLS_cdrViewType')} **Related Topics**

cdrWindowArrangeStyle

Enum **cdrWindowArrangeStyle**

Referenced By

Constant	Value	Description
cdrTileHorizontally	0	
cdrTileVertically	1	
cdrCascade	2	

{button ,AL(`CLS_cdrWindowArrangeStyle')} **Related Topics**

cdrWindowState

Enum **cdrWindowState**

Referenced By

Constant	Value	Description
cdrWindowNormal	1	
cdrWindowMaximized	3	
cdrWindowMinimized	2	
cdrWindowRestore	9	

{button ,AL(`CLS_cdrWindowState')} **Related Topics**

cuiBarPosition

Enum **cuiBarPosition**

Referenced By

Enumerated BarPosition values

Constant	Value	Description
cuiBarLeft	0	
cuiBarTop	1	
cuiBarRight	2	
cuiBarBottom	3	
cuiBarFloating	4	

{button ,AL(`CLS_cuiBarPosition')} **Related Topics**

cuiBarProtection

Enum **cuiBarProtection**

Referenced By

Enumerated BarProtection values

Constant	Value	Description
cuiBarNoProtection	0	
cuiBarNoCustomize	1	
cuiBarNoMove	4	
cuiBarNoChangeVisible	8	
cuiBarNoChangeDock	16	
cuiBarNoVerticalDock	32	
cuiBarNoHorizontalDock	64	

{button ,AL(`CLS_cuiBarProtection')} **Related Topics**

cuiBarType

Enum **cuiBarType**

Referenced By

Enumerated BarType values

Constant	Value	Description
cuiBarTypeNormal	0	
cuiBarTypeMenuBar	1	
cuiBarTypePopup	2	
cuiBarTypeStatusBar	3	
cuiBarTypePropertyBar	4	

{button ,AL(`CLS_cuiBarType')} **Related Topics**

pdfBitmapCompressionType

Enum **pdfBitmapCompressionType**

Referenced By

Constant	Value	Description
pdfNone	0	
pdfLZW	1	
pdfJPEG	2	
pdfZIP	3	

{button ,AL(`CLS_pdfBitmapCompressionType')} **Related Topics**

pdfColorMode

Enum **pdfColorMode**

Referenced By

Constant	Value	Description
pdfRGB	0	
pdfCMYK	1	
pdfGrayscale	2	
pdfNative	3	

{button ,AL(`CLS_pdfColorMode')} **Related Topics**

pdfColorProfile

Enum **pdfColorProfile**

Referenced By

Constant	Value	Description
pdfCompositeProfile	0	
pdfSeparationProfile	1	

{button ,AL(`CLS_pdfColorProfile')} **Related Topics**

pdfDisplayOnStart

Enum **pdfDisplayOnStart**

Referenced By

Constant	Value	Description
pdfPageOnly	0	
pdfFullScreen	1	
PDFBookmarks	2	
pdfThumbnails	3	

{button ,AL(`CLS_pdfDisplayOnStart')} **Related Topics**

pdfEncodingType

Enum **pdfEncodingType**

Referenced By

Constant	Value	Description
pdfASCII85	0	
pdfBinary	1	

{button ,AL(`CLS_pdfEncodingType')} **Related Topics**

pdfEPSAs

Enum **pdfEPSAs**

Referenced By

Constant	Value	Description
pdfPostscript	0	
pdfPreview	1	

{button ,AL(`CLS_pdfEPSAs')} **Related Topics**

pdfExportRange

Enum **pdfExportRange**

Referenced By

Constant	Value	Description
pdfWholeDocument	0	
pdfCurrentPage	1	
pdfSelection	2	
pdfPageRange	3	

{button ,AL(`CLS_pdfExportRange')} **Related Topics**

pdfVersion

Enum **pdfVersion**

Referenced By

Constant	Value	Description
pdfVersion12	0	
pdfVersion13	1	
pdfVersionPDFX1	2	

{button ,AL(`CLS_pdfVersion')} **Related Topics**

PrnBitmapColorMode

Enum **PrnBitmapColorMode**

Referenced By

Constant	Value	Description
prnBitmapCMYK	0	
prnBitmapRGB	1	
prnBitmapGrayscale	2	

{button ,AL(`CLS_PrnBitmapColorMode')} **Related Topics**

PrnColorMode

Enum **PrnColorMode**

Referenced By

Constant	Value	Description
prnModeFullColor	0	
prnModeGrayscale	1	
prnModeBlack	2	

{button ,AL(`CLS_PrnColorMode')} **Related Topics**

PrnFileMode

Enum **PrnFileMode**

Referenced By

Constant	Value	Description
prnSingleFile	0	
prnSeparatePages	1	
prnSeparatePlates	2	

{button ,AL(`CLS_PrnFileMode')} **Related Topics**

PrnImageTrap

Enum **PrnImageTrap**

Referenced By

Constant	Value	Description
prnTrapNormal	0	
prnTrapSpread	1	
prnTrapChoke	2	
prnTrapCenter	3	

{button ,AL(`CLS_PrnImageTrap')} **Related Topics**

PrnPDFStartup

Enum **PrnPDFStartup**

Referenced By

Constant	Value	Description
prnPDFFullScreen	0	
prnPDFPageOnly	1	
prnPDFThumbnails	2	
prnPDFOutlines	3	

{button ,AL(`CLS_PrnPDFStartup')} **Related Topics**

PrnPlateType

Enum **PrnPlateType**

Referenced By

Constant	Value	Description
prnCyan	0	
prnMagenta	1	
prnYellow	2	
prnBlack	3	
prnOrange	4	
prnGreen	5	
prnSpot	6	

{button ,AL(`CLS_PrnPlateType')} **Related Topics**

PrnPostScriptLevel

Enum **PrnPostScriptLevel**

Referenced By

Constant	Value	Description
prnPSLevel1	1	
prnPSLevel2	2	
prnPSLevel3	3	

{button ,AL(`CLS_PrnPostScriptLevel')} **Related Topics**

PrnPrintRange

Enum **PrnPrintRange**

Referenced By

Constant	Value	Description
prnWholeDocument	0	
prnCurrentPage	1	
prnSelection	2	
prnPageRange	3	

{button ,AL(`CLS_PrnPrintRange')} **Related Topics**

PrnRegistrationStyle

Enum **PrnRegistrationStyle**

Referenced By

Constant	Value	Description
prnStandard	0	
prnLong	1	
prnSquare	2	
prnHalfInverted	3	
prnCorel	4	

{button ,AL(`CLS_PrnRegistrationStyle')} **Related Topics**

PrnTrapType

Enum **PrnTrapType**

Referenced By

Constant	Value	Description
prnLayerNormal	0	
prnLayerTransparent	1	
prnLayerOpaque	2	
prnLayerOpaqueIgnore	3	

{button ,AL(`CLS_PrnTrapType')} **Related Topics**

ANGLECONVERT function

ANGLECONVERT(x, y, z)

Converts a number from one angle measurement to another.

Parameter	Description
x	Any number from 1 to 5 that indicates the unit of measurement from which to convert. 1 = degrees 2 = radians 3 = gradients 4 = Corel PHOTO-PAINT degrees (tenths of a degree) 5 = CorelDRAW degrees (millionths of a degree)
y	Any number from 1 to 5 that indicates the unit of measurement to convert to. 1 = degrees 2 = radians 3 = gradients 4 = Corel PHOTO-PAINT degrees (tenths of a degree) 5 = CorelDRAW degrees (millionths of a degree)
z	Any numeric expression specifying the value to be converted.

Example

x_rads = ANGLECONVERT(1, 2, 90)

The above example converts 90 degrees to radians. The variable **x_rads** equals 1.57142857142932.

{button ,AL(`include;cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMCENTIMETERS function

FROMCENTIMETERS(x)

Converts a numeric value from centimeters to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMCENTIMETERS(8), FROMCENTIMETERS(-5), FROMCENTIMETERS(0), FROMCENTIMETERS(2.5), FROMCENTIMETERS(0.75)

This CorelDRAW command would create a rectangle 7.5 by 8 centimeters. The rectangle's top left corner coordinate is -5, 8 centimeters relative to the center of the page, and the corners are 0.75 centimeters in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMCICEROS function

FROMCICEROS(x)

Converts a numeric value from ciceros to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMCICEROS(18), FROMCICEROS(-12), FROMCICEROS(8), FROMCICEROS(6), FROMCICEROS(1.5)

This CorelDRAW command would create a rectangle 18 by 10 ciceros. The rectangle's top left corner coordinate is -12, 18 ciceros relative to the center of the page, and the corners are 1.5 ciceros in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMDIDOTS function

FROMDIDOTS(x)

Converts a numeric value from didots to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMDIDOTS(50), FROMDIDOTS(-70), FROMDIDOTS(0), FROMDIDOTS(30), FROMDIDOTS(20)

This CorelDRAW command would create a rectangle 100 by 50 didots. The rectangle's top left coordinate is -70, 50 didots relative to the center of the page, and the corners are 20 didots in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMINCHES function

FROMINCHES(x)

Converts a numeric value from inches to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMINCHES(3), FROMINCHES(-2), FROMINCHES(0), FROMINCHES(1), FROMINCHES(0.25)

This CorelDRAW command would create a rectangle 3 by 3 inches. The rectangle's top left coordinate is -2, 3 inches relative to the center of the page, and the corners are 0.25 inches in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMPICAS function

FROMPICAS(x)

Converts a numeric value from picas to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMPICAS(18), FROMPICAS(-12), FROMPICAS(8), FROMPICAS(6), FROMPICAS(2)

This CorelDRAW command would create a rectangle 18 by 10 picas. The rectangle's top left coordinate is -12, 18 picas relative to the center of the page, and the corners are 2 picas in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

FROMPOINTS function

FROMPOINTS(x)

Converts a numeric value from points to tenths of a micron.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.CreateRectangle FROMPOINTS(210), FROMPOINTS(-140), FROMPOINTS(90), FROMPOINTS(70), FROMPOINTS(1.75)

This CorelDRAW command would create a rectangle 210 by 140 points. The rectangle's top left corner coordinate is -140, 210 points relative to the center of the page, and the corners are 1.75 points in diameter.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

LENGTHCONVERT function

LENGTHCONVERT(x, y, z)

Converts a number from one length measurement to another.

Parameter	Description
x	Any number from 1 to 7 that indicates the unit of measurement from which to convert. 1 inches 2 centimeters 3 points 4 Ciceros 5 didots 6 picas 7 CorelDRAW and VENTURA units (tenths of a micron)
y	Any number from 1 to 7 that indicates the unit of measurement to convert to: 1 inches 2 centimeters 3 points 4 Ciceros 5 didots 6 picas 7 CorelDRAW and VENTURA units (tenths of a micron)
z	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

x_microns = LENGTHCONVERT(1, 7, 1)

The above example converts one inch to tenths of a micron. The variable **x_microns** equals 254,000.

{button ,AL(`include;cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TOCENTIMETERS function

TOCENTIMETERS(x)

Converts a numeric value from tenths of a micron to centimeters.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xCM = TOCENTIMETERS (x)

yCM = TOCENTIMETERS (y)

In this CorelDRAW example, **xCM** and **yCM** are set to the X and Y coordinates of the selected object in centimeters.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TOCICEROS function

TOCICEROS(x)

Converts a numeric value from tenths of a micron to ciceros.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xCiceros = TOCICEROS (x)

yCiceros = TOCICEROS (y)

In this CorelDRAW example, **xCiceros** and **yCiceros** are set to the X and Y coordinates of the selected object in ciceros.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TODIDOTS function

TODIDOTS(x)

Converts a numeric value from tenths of a micron to didots.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xDidots = TODIDOTS (x)

yDidots = TODIDOTS (y)

In this CorelDRAW example, **xDidots** and **yDidots** are set to the X and Y coordinates of the selected object in didots.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TOINCHES function

TOINCHES(x)

Converts a numeric value from tenths of a micron to inches.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xInch = TOINCHES (x)

yInch = TOINCHES (y)

In this CorelDRAW example, **xInch** and **yInch** are set to the X and Y coordinates of the selected object in inches.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TOPICAS function

TOPICAS(x)

Converts a numeric value from tenths of a micron to picas.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xPica = TOPICAS (x)

yPica = TOPICAS (y)

In this CorelDRAW example, **xPica** and **yPica** are set to the X and Y coordinates of the selected object in picas.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

TOPOINTS function

TOPOINTS(x)

Converts a numeric value from tenths of a micron to points.

Parameter	Description
x	Any numeric expression specifying the value to be converted.

Note

- Tenths of a micron are used as the basic unit of measurement in Corel applications such as CorelDRAW and Corel VENTURA.

Example

.GetPosition (x,y)

xPoint = TOPOINTS (x)

yPoint = TOPOINTS (y)

In this CorelDRAW example, **xPoint** and **yPoint** are set to the X and Y coordinates of the selected object in points.

{button ,AL(`cs_converts;;;;;',0,"Defaultoverview",)} Related Topics

A

AddArrowPoint
AddBezierPoint
AddFreehandPoint
AddNode
AddPageFrame
AddTabStop
AfterObject
AlignObjects
AlignTextToBaseline
AlignToCenterOfPage
AlignToGrid
AppendCurveLine
AppendObjectToSelection
ApplyBitmapEffect
ApplyBlend
ApplyContour
ApplyDistortion
ApplyDropShadow
ApplyEnvelopeFrom
ApplyExtrude

ApplyFountainFill
ApplyFullColorFill
ApplyLensEffect
ApplyNoFill
ApplyOutline
ApplyPerspectiveEffect
ApplyPostscriptFill
ApplyPreset
ApplyPresetEnvelope
ApplyRotatedExtrude
ApplyStyle
ApplyTextureFill
ApplyTwoColorFill
ApplyUniformFillColor

B

BeforeObject
BeginCommandGroup
BeginDrawArrow
BeginDrawBezier
BeginDrawCurve
BeginDrawFreehand
BeginEditObject
BreakApart

C

ChangeLayerName
ChangeLayerColor
ClearEffect
ClearNodeSelection
ClickedDialogButton
CloneObject
CloseCurve
Combine
ConvertBitmapTo
ConvertColor
ConvertOutlineToObject
ConvertToBitmap
ConvertToCurves
CopyEffectFrom
CopyPowerClip
CopyPropertiesFrom
CopyToClipboard
CopyToLayer
CreateAngleDimension
CreateArtisticText
CreateCallout
CreateConnector
CreateDimension
CreateEllipse
CreateGridBoxes

CreateGuidelineUsingAngle
CreateGuidelineUsingTwoPoints
CreatePaletteFromDocument
CreatePaletteFromSelection
CreateRectangle
CreateSymPolygon
CreateTextString
CutToClipboard

D

DeleteGuidelineByIndex
DeleteGuidelineUsingAngle
DeleteGuidelineUsingTwoPoints
DeleteLayer
DeleteNode
DeleteObject
DeletePages
DeletePaletteColor
DeleteStyle
DetachBlendPath
DisplayFacingPages
DistributeObjects
DrawCurveClosePath
DrawCurveCurveTo
DrawCurveLineTo
DrawCurveMoveTo
DropSymbol
DuplicateObject

E

EditAngleDimensionLabel
EditDimensionLabel
EditLayer
EditObjectCommand
EndCommandGroup
EndDrawArrow
EndDrawBezier
EndDrawCurve
EndDrawFreehand
EndEditObject
EndOfRecording
ExtractContents
ExtractText

F

FileClose
FileExit
FileExport
FileImport
FileNew
FileOpen

FilePrint
FileSave
FindNextObjectOfStyle
FindObjectOfStyle
FitTextToPath
FuseBlend

G

GetBitmapResolution
GetBitmapSize
GetCDRFileCompRatio
GetCDRFileKeywords
GetCDRFileLastSavedBy
GetCDRFileNotes
GetCDRFileThumbnail
GetCDRFileVersion
GetCurrentPageName
GetCurrentPageOrientation
GetCurrentPageSize
GetCurrentPaletteName
GetCurrentWorkspaceName
GetCurveClose
GetCurveFirstNodePosition
GetCurveIthNodePosition
GetCurveLastNodePosition
GetCurveLength
GetCurveNodeCount
GetCurveSubpathCount
GetDocumentCount
GetDocumentName
GetEllipseClockwise
GetEllipseEndAngle
GetEllipseStartAngle
GetEllipseType
GetFillType
GetFountainFillColor
GetFountainFill
GetGuidelineInformation
GetNodeIndex
GetNodePosition
GetNodeSelectedCount
GetNodeType
GetNumberOfGuidelines
GetObjectCount
GetObjectID
GetObjectsCDRStaticID
GetObjectType
GetOutlineColor
GetOutline
GetPageCount

GetPageSize
GetPaletteColor
GetPaletteColorName
GetPolygonSharpness
GetPolygonSides
GetPolygonType
GetPosition
GetRectangleRadius
GetSegmentLength
GetSegmentType
GetSize
GetTextFontName
GetTextFontSize
GetTextString
GetTextWordCount
GetUniformFillColor
GetUserClick
GetUserDataField
GetWorkspaceCount
GetWorkspaceDescription
GetWorkspaceName
Group

H

I

InflateBitmap
InitBezierTool
InsertOLEObjectFromFile
InsertOLEObject
InsertPages
InsertPaletteColor
Intersection
IsBitmapExternallyLink
IsDefaultWorkspace
IsDocument
IsSelection

J

K

L

LoadPalette
LoadStyles
LockGuidelineByIndex

M

MenuCommand
MergeBackText
MoveBezierControl
MoveCenter
MoveGuidelineUsingAngleByIndex

MoveGuidelineUsingTwoPointsByIndex
MoveLayerTo
MoveNode
MoveObject
MoveToLayer

N

NewLayer

O

OLEObjectDoVerb
OrderBackOne
OrderForwardOne
OrderReverseOrder
OrderToBack
OrderToFront
OverPrintFill
OverPrintOutline

P

PasteCustomClipboardFormat
PasteFromClipboard
PasteSystemClipboardFormat
PlaceInside

Q

R

RecorderApplyPerspective
RecorderBeginEditParaText
RecorderBeginEditText
RecorderEditParaTextChangeCase
RecorderEditParaTextCharAttributes
RecorderEditParaTextIndents
RecorderEditParaReplaceText
RecorderEditParaTextSpacing
RecorderEditTextChangeCase
RecorderEditTextCharAttributes
RecorderEditReplaceText
RecorderEndEditParaText
RecorderEndEditText
RecorderObjectScaleInfo
RecorderSelectObjectByIndex
RecorderSelectObjectsByIndex
RecorderSelectPreselectedObjects
RecorderStorePreselectedObjects
Redo
RedrawAllScreens
RedrawScreen
RemoveAllGuidelines
RemoveFountainFillColor
RepeatLastCommand

Repeat
ReplacePaletteColor
ResetTransfo
ResolveAllBitmapsLink
ResolveBitmapLink
ResumePainting
RevertToStyle
RotateObject

S

SavePalette
SaveStyleAs
SaveTemplate
SelectAllObjects
SelectLayer
SelectNextNode
SelectNextObject
SelectNode
SelectNodeAt
SelectObjectAtPoint
SelectObjectOfCDRStaticID
SelectObjectOfType
SelectObjectsInRect
SelectPreviousObject
Separate
SetApplyToDuplicate
SetArtisticText
SetBullet
SetCharacterAttributes
SetColorOverride
SetCornerRoundness
SetCurrentDocument
SetCurrentPage
SetCurrentPageName
SetCurrentPageOrientation
SetCurrentPageSize
SetCurrentWorkspace
SetDocVisible
SetEllipseProperties
SetErrorHandling
SetFrameColumn
SetFullScreenPreview
SetIndents
SetLayerLocked
SetLayerPrintable
SetLayerVisible
SetNodeType
SetMultiLayer
SetOptionsForAllPages
SetOutlineArrow

SetOutlineColor
SetOutlineMiscProperties
SetOutlineWidth
SetPageLayout
SetPageOrientation
SetPageSizeFromPrinter
SetPageSize
SetPaperColor
SetParagraphSpacing
SetPolygonProperties
SetPosition
SetReferencePoint
SetSegmentType
SetSize
SetTextString
SetToMasterLayer
SetUserDataField
SetVisible
ShareExtrudeVP
ShowPageBorder
SkewObject
SplitBlend
StartEditContents
StartOfRecording
StopEditContents
StoreColor
StraightenText
StretchObject
SuppressPainting

T

Trim

U

Undo
Ungroup
UngroupAll
UnlockGuidelineByIndex
UnSelectAll
UpdateBitmapLink

V

W

X

Y

Z

ZoomIn
ZoomOut
ZoomToAllObjects

ZoomToHeight
ZoomToPage
ZoomToRectangle
ZoomToSelection
ZoomToWidth

File commands

FileClose (DRAW)

ReturnValue = .FileClose(.PromptUser = *boolean*)

This command closes the current drawing.

Return Value

Returns one of the following values:

- TRUE (-1)▸the file was closed
- FALSE (0)▸the file was not closed

Parameter	Description
.PromptUser	Set to TRUE (-1) to prompt the user before closing the file. Set to FALSE (0) to close the file without prompting the user.

 **Note**

- This command must be preceded by the .FileSave command or changes will be lost.

Example

```
.FileClose TRUE
```

The above example prompts the user before closing the active CorelDRAW document.

```
.FileClose
```

The above example closes the active CorelDRAW document without prompting the user.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

FileExit (DRAW)

ReturnValue = .FileExit .PromptUser = *boolean*

This command exits the application.

Return Value

Returns one of the following values:

- TRUE (-1)▸the application was closed
- FALSE (0)▸the application was not closed.

Parameter	Description
.PromptUser	Set to TRUE (-1) to prompt the user before closing the application. Set to FALSE (0) to exit the application without prompting the user.

▸Note

- This command must be preceded by the .FileSave command or changes will be lost.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

FileExport (DRAW)

.FileExport .FileName = *string*, .FilterID = *long*, .Width = *long*, .Height = *long*, .XResolution = *long*, .YResolution = *long*, .ImageType = *long;* Antialiasing = *long;* Overwrite = *boolean;* SelectionOnly = *boolean*

This command saves the current drawing in a format that other programs can read.

Parameter	Description
.FileName	Lets you specify the name of the file to export.
.FilterID	Lets you specify the type of file filter. 769 = Windows Bitmap (BMP) 770 = Paintbrush (PCX) 771 = Targa Bitmap (TGA) 772 = TIFF Bitmap (TIF) 773 = CompuServe Bitmap (GIF) 774 = JPEG Bitmap (JPG) 776 = Scitex CT Bitmap (SCT) 787 = GEM Paint File (IMG) 788 = Adobe Photoshop (PSD) 791 = MACPaint Bitmap (MAC) 792 = OS/2 Bitmap (BMP) 800 = CALS Compressed Bitmap (CAL) 802 = Portable Network Graphics (PNG) 806 = Kodak FlashPix Image (FPX) 1280 = Computer Graphics Metafile (CGM) 1281 = HPGL Plotter File (PLT) 1282 = Micrografx 2.x, 3.x (DRW) 1284 = GEM File (GEM) 1285 = IBM PIF (PIF) 1287 = WordPerfect Graphics (WPG) 1289 = Encapsulated PostScript (EPS) 1293 = Macintosh Pict (PCT) 1294 = Windows Metafile (WMF) 1296 = AutoCad (DXF) 1298 = Scodl (SCD) 1300 = Enhanced Windows Metafile (EMF) 1302 = True Type Font (TTF) 1303 = Adobe Type 1 Font (PFB) 1305 = Adobe Illustrator (AI) 1312 = Corel Barista (HTM) 1313 = Corel Image Map (HTM) 1329 = Frame Vector Metafile (FMV) 1333 = Adobe Portable Document File (PDF) 1792 = Corel PHOTO-PAINT Image (CPT) Ver 5.0/6.0 1793 = Corel Presentation Exchange 6/7 (CMX) 1794 = Corel Presentation Exchange 5.0 (CMX) 1799 = Corel PHOTO-PAINT Image (CPT) 2048 = ANSI Text (TXT) 2049 = MS Word for Windows 6/7 (DOC) 2050 = MS Word for Windows 2.x (DOC) 2051 = MS Word 4.0, 5.0, 5.5 (DOC) 2052 = MS Word for Macintosh 4.0, 5.0 (DOC) 2053 = Rich Text Format (RTF) 2055 = Corel WordPerfect 6/7/8 (WPD) 2056 = Corel WordPerfect 5.1 (WP5) 2057 = Corel Word Perfect 5.0 (WP5) 2058 = Corel Word Perfect 4.2 (WP5) 2059 = Word Star for Windows 1.x, 2.0 (WSW) 2060 = Word Star 7.0 (WSD) 2061 = Word Star 2000 (WSD) 2062 = XYWrite for Windows (XY) 2068 = MS Word 97 (DOC)
.Width	Lets you specify the width of the image in pixels.
.Height	Lets you specify the height of the image in pixels.
.XResolution	Lets you specify the horizontal resolution of the image in dots per inch (dpi).
.YResolution	Lets you specify the vertical resolution of the image in dots per inch (dpi).

.ImageType	Lets you specify the image type. 1 = Monochrome bitmap 3 = 8-bit paletted color bitmap 4 = 24-bit RGB color bitmap 6 = 32-bit CMYK bitmap 10 = 4-bit, 16 colors (standard VGA palette)
.Antialiasing	0 = None 1 = Normal 2 = Super-Sampling
.Overwrite	When set to TRUE (-1) overwrites the file if one exists.
.SelectionOnly	When set to TRUE (-1) exports only the current selection.

Example

```
.FileExport "C:\TEMP1.BMP", 769, 320, 400, 72, 72, 4
```

The above example exports a CorelDRAW file to a Windows bitmap named "TEMP1.BMP".

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

FileImport (DRAW)

.FileImport .FileName = *string,* .FilterID = *long,* MaintainLayersAndPages = *boolean*

This command brings graphics into CorelDRAW from other programs.

Parameter	Description
.FileName	Lets you specify the name of the file to import.
.FilterID	Lets you specify the type of file filter: 769 = Windows Bitmap (BMP) 770 = Paintbrush (PCX) 771 = Targa Bitmap (TGA) 772 = TIFF Bitmap (TIF) 773 = CompuServe Bitmap (GIF) 774 = JPEG Bitmap (JPG) 775 = Kodak Photo CD Image (PCD) 776 = Scitex CT Bitmap (SCT) 784 = Windows 3.x/NT Cursor resource (CUR) 785 = Windows 3.x/NT Icon resource (ICO) 786 = Windows 3.x/NT Bitmaps resource (EXE) 787 = GEM Paint File (IMG) 788 = Adobe Photoshop (PSD) 789 = Picture Publisher 4 (PP4) 791 = MACPaint Bitmap (MAC) 792 = OS/2 Bitmap (BMP) 793 = Wavelet Compressed Bitmaps (WVL) 800 = CALS Compressed Bitmap (CAL) 802 = Portable Network Graphics (PNG) 803 = Picture Publisher 5.0, 6 (PP5, PP6) 806 = Kodak FlashPix Image (FPX) 1280 = Computer Graphics Metafile (CGM) 1281 = HPGL Plotter File (PLT) 1282 = Micrografx 2.x, 3.x (DRW) 1284 = GEM File (GEM) 1285 = IBM PIF (PIF) 1287 = WordPerfect Graphics (WPG) 1289 = Encapsulated PostScript (EPS) 1290 = PostScript Interpreted (PS, PRN, ES) 1291 = MET Maetfile (MET) 1292 = NAP Metafile (NAP) 1293 = Macintosh Pict (PCT) 1294 = Windows Metafile (WMF) 1296 = AutoCad (DXF) 1298 = Scodl (SCD) 1300 = Enhanced Windows Metafile (EMF) 1302 = True Type Font (TTF) 1303 = Adobe Type 1 Font (PFB) 1305 = Adobe Illustrator (AI) 1312 = Corel Barista (HTM) 1313 = Corel Image Map (HTM) 1314 = Placeable Enhanced PDF (PDF) 1315 = Visio (VSD) 1329 = Frame Vector Metafile (FMV) 1333 = Adobe Portable Document File (PDF) 1334 = Lotus Pic (PIC) 1339 = Micrografx Designer 6.0 (DSF) 1557 = HyperText Markup Language (HTM) 1792 = Corel PHOTO-PAINT Image (CPT) Ver 5.0/6.0 1793 = Corel Presentation Exchange 6/7 (CMX) 1794 = Corel Presentation Exchange 5.0 (CMX) 1796 = CorelDRAW Compressed (CDX) 1797 = Corel CMX Compressed (CPX) 1799 = Corel PHOTO-PAINT Image (CPT) 2048 = ANSI Text (TXT) 2049 = MS Word for Windows 6/7 (DOC) 2050 = MS Word for Windows 2.x (DOC) 2051 = MS Word 4.0, 5.0, 5.5 (DOC) 2052 = MS Word for Macintosh 4.0, 5.0 (DOC)

2053 = Rich Text Format (RTF)
2055 = Corel WordPerfect 6/7/8 (WPD)
2056 = Corel WordPerfect 5.1 (WP5)
2057 = Corel Word Perfect 5.0 (WP5)
2058 = Corel Word Perfect 4.2 (WP5)
2059 = Word Star for Windows 1.x, 2.0 (WSW)
2060 = Word Star 7.0 (WSD)
2061 = Word Star 2000 (WSD)
2062 = XYWrite for Windows (XY)
2063 = Ami Professional 2.0, 3.0 (SAM)
2068 = MS Word 97 (DOC)

.MaintainLayersAndPages — When set to TRUE (-1) maintain layers and pages during import.

Example

```
.FileNew
.FileImport "C:\TEST1.BMP", 769, FALSE
```

The above example creates a new document and imports a Windows bitmap file named "TEST1.BMP" into the document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

FileNew (DRAW)

ReturnValue = .FileNew

This command creates a new drawing.

Return Value

Returns one of the following values:

- TRUE (-1) the file was created
- FALSE (0) the file was not created

Note

- You cannot change the active CorelDRAW document in a script except by using the .FileNew or .FileOpen command.

  Changing the active CorelDRAW document with keyboard and mouse actions does not affect an executing script.

Example

```
.FileNew
```

The above example creates a new CorelDRAW document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

FileOpen (DRAW)

.FileOpen .FileName = *string*

This command loads a drawing or Styles template into CorelDRAW.

Parameter	Description
.FileName	Lets you specify the name of the file to open.

Note

- You cannot change the active CorelDRAW document in a script except by using the .FileNew or .FileOpen command. Changing the active CorelDRAW document with keyboard and mouse actions does not affect an executing script.

Example

```
.FileOpen "C:\TEST1.CDR"
```

The above example opens a CorelDRAW file named "TEST1.CDR".

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

FilePrint (DRAW)

.FilePrint

This command prints the active document.

Example

```
.FilePrint
```

The above example sends the active document to the printer.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

FileSave (DRAW)

.FileSave .FileName = *string*, .ThumbNailSize = *long*, .SaveSelectedOnly = *boolean*, .FileVersion = *long*, .IncludeCMXData = *boolean*

This command saves the active document.

Parameter	Description
.FileName	Lets you specify the name of the file to save.
.ThumbNailSize	Lets you specify the size of the thumbnail: 0 = None 1 = 1k (mono) 5 = 5k (color) 10 = 10k (color)
.SaveSelectedOnly	Set to TRUE (-1) to save selected items only. Set to FALSE (0) to save entire document.
.FileVersion	Lets you specify the file version of the document being saved. 0 = Version 9.0 1 = Version 5.0 2 = Version 6.0 3 = Version 7.0 4 = Version 8.0
.IncludeCMXData	Set to TRUE (-1) to include CMX data with the saved file. Set to FALSE (0) to disable this feature.

Example

```
.FileSave "C:\TEST1.CDR", 1, 0, 0, 0
```

The above example saves a version 9 CorelDRAW document named "TEST1.CDR", with a 1k thumbnail. CMX data is not saved.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

GetCDRFileKeywords (DRAW)

.GetCDRFileKeywords .Filename = *filename*

This command returns the keywords of the current document.

Parameter	Description
.Filename	Returns the keywords for the current document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCDRFileCompRatio (DRAW)

.GetCDRFileCompRatio .Filename = *filename*

This command returns the file compression ratio of the current document.

Parameter	Description
.Filename	Returns the file compression ratio for the current document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCDRFileLastSavedBy (DRAW)

.GetCDRFileLastSavedBy .Filename = *filename*

This command returns the who saved the file last for the current document.

Parameter	Description
.Filename	Returns the last saved by name for the current document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCDRFileNotes (DRAW)

.GetCDRFileNotes .Filename = *filename*

This command returns the notes of the current document.

Parameter	Description
.Filename	Returns the notes for the current document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} Related Topics

GetCDRFileThumbnail (DRAW)

.GetCDRFileThumbnail .CDRFilename = *filename,* .BMPFilename = *filename*

This command extracts the thumbnail of the CorelDRAW file and returns a BMP file.

Parameter	Description
.CDRFilename	Lets you specify the CorelDRAW file name from which to extract the thumbnail.
.BMPFilename	Returns the thumbnail image for the specified CorelDRAW file name.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCDRFileVersion (DRAW)

.GetCDRFileVersion .Filename = *filename*

This command returns the file version of the current document.

Parameter	Description
.Filename	Returns the file name for the current document.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCurrentWorkspaceDescription (DRAW)

.GetCurrentWorkspaceDescription .Name = *string*

This command returns the description for a specifies workspace.

Parameter	Description
.Name	Lets you specify the workspace name for which you want a description.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetCurrentWorkspaceName (DRAW)

.GetCurrentWorkspaceName

This command returns the current workspace name.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetWorkspaceCount (DRAW)

.GetWorkspaceCount

This command returns the number of workspaces.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetWorkspaceDescription (DRAW)

.GetWorkspaceDescription .Name = *string*

This command returns the workspace description.

Parameter	Description
.Name	Lets you specify the workspace name for which you want the description.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

GetWorkspaceName (DRAW)

.GetWorkspaceName .Index = *long*

This command returns the name of the I-th workspace.

Parameter	Description
.Index	Lets you specify the workspace index number for which you want the name.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

IsDefaultWorkspace (DRAW)

.IsDefaultWorkspace .Name = *string*

This command specifies the default workspace.

Parameter	Description
.Name	Lets you specify the workspace name that you want to make the default workspace.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

IsDocument (DRAW)

This command returns true if a document is open.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

SetCurrentWorkspace (DRAW)

.SetCurrentWorkspace .Name = *string*

This command saves the current CorelDRAW workspace settings.

Parameter	Description
.Name	Lets you specify the workspace name to save for the current settings.

{button ,AL(`OVR1 File commands;',0,"Defaultoverview",)} **Related Topics**

Edit commands

BeginCommandGroup (DRAW)

.BeginCommandGroup .UndoString = *string*

This command starts a group of commands to have a clean undo stack.

Parameter	Description
.UndoString	Lets you specify the string to be used.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} **Related Topics**

CopyPropertiesFrom (DRAW)

.CopyPropertiesFrom .FromObjectID = *long*, .OutlinePen = *boolean*, .OutlineColor = *boolean*, .Fill = *boolean*, .TextAttibutes = *boolean*

This command copies the properties from the object with the specified object ID to the selected object.

Parameter	Description
.FromObjectID	Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.
.OutlinePen	Set to TRUE (-1) to copy outline pen properties. Set to FALSE (0) to exclude outline pen properties.
.OutlineColor	Set to TRUE (-1) to copy outline color properties. Set to FALSE (0) to exclude outline color properties.
.Fill	Set to TRUE (-1) to copy fill properties. Set to FALSE (0) to exclude fill properties.
.TextAttibutes	Set to TRUE (-1) to copy text properties. Set to FALSE (0) to exclude text properties.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} **Related Topics**

CopyToClipboard (DRAW)

.CopyToClipboard

This command places a copy of the selected object(s) or text onto the Clipboard.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 2
.PasteFromClipboard
```

The above example copies a rectangle to the Clipboard, inserts 2 pages, then pastes the contents of the Clipboard to the third page.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

CutToClipboard (DRAW)

.CutToClipboard

This command removes the selected object(s) or text from your document and places a copy onto the Clipboard.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

EndCommandGroup (DRAW)

.EndCommandGroup

This command ends a group of commands.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} **Related Topics**

InsertOLEObject (DRAW)

.InsertOLEObject .ProgID = *string*

This command inserts an OLE object in a CorelDRAW document.

Parameter	Description
.ProgID	Lets you specify the OLE object's Windows registry name.

Example

```
.InsertOLEObject "CorelPhotoPaint.Image.6"
```

The above example inserts a Corel PHOTO-PAINT image into a CorelDRAW document.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

InsertOLEObjectFromFile (DRAW)

.InsertOLEObjectFromFile .FileName = *string*, .CreateLink = *boolean*

This command inserts an OLE object from a file into a CorelDRAW document.

Parameter	Description
.FileName	The filename.
.CreateLink	Set to TRUE (-1) to create a link. Set to FALSE (0) to disable this option.

Example

```
.InsertOLEObjectFromFile "C:\WINWORD\WORDFILE.DOC", -1
```

The above example inserts a Microsoft Word file in a CorelDRAW document.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

OLEObjectDoVerb (DRAW)

.OLEObjectDoVerb .Verb = *long*

This command performs the specified action on an OLE object.

Parameter	Description
.Verb	Lets you specify the OLE object action to perform. 0 = Primary 1 = Secondary 2 = Tertiary etc.

Note

- Primary and secondary verbs depend on the object type.

Example

```
.InsertOLEObject "CorelPhotoPaint.Image.7"
.OLEObjectDoVerb 0
```

The above example inserts a Corel PHOTO-PAINT OLE object into a DRAW document and invokes in-place editing.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

PasteCustomClipboardFormat (DRAW)

.PasteCustomClipboardFormat .Format = *string*

This command specifies the custom format for pasting from the Clipboard.

Parameter	Description
.Format	Lets you specify the type of format. Options include: "Corel 32-bit Presentation Exchange Data" "Corel Presentation Exchange Data" "Corel Metafile" "Rich Text Format"

Example

```
.PasteCustomClipboardFormat "Rich Text Format"
```

The above example inserts the contents of the Clipboard into a CorelDRAW document as Rich Text.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

PasteFromClipboard (DRAW)

.PasteFromClipboard

This command places a copy of the object(s) on the Clipboard into your drawing.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 2
.PasteFromClipboard
```

The above example copies a rectangle to the Clipboard, inserts 2 pages, then pastes the contents of the Clipboard in to the last page inserted.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

PasteSystemClipboardFormat (DRAW)

.PasteSystemClipboardFormat .Format = *long*

This command specifies the system format for pasting from the Clipboard.

Parameter	Description
.Format	Lets you specify the type of format. 1 = CF Text 2 = Bitmap 3 = Metafile Pict 8 = DIB 14 = Enhanced Metafile

Example

```
.PasteSystemClipboardFormat 2
```

The above example pastes a bitmap from the Clipboard into the active document.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

Redo (DRAW)

.Redo

This command restores changes reversed by the Undo command. Redo becomes available immediately after you select the Undo command.

Example

```
.Redo
```

The above command reverses the last .Undo command and reinstates the previous deletion or reversal of actions.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

Repeat (DRAW)

.Repeat

This command applies, if possible, the most recent command or action to selected object.

Example

```
.Repeat
```

The above example repeats the last command.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

RepeatLastCommand (DRAW)

.RepeatLastCommand

This command applies, if possible, the last command or action to selected object.

Example

```
.RepeatLastCommand
```

The above example repeats the last command.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

SetErrorHandling (DRAW)

.SetErrorHandling .Msg = *boolean*

This command sets how the errors should be returned.

Parameter	Description
.Msg	Set to TRUE (-1) returns error messages. Set to FALSE (0) returns error code.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

Undo (DRAW)

.Undo

This command reverses actions performed during the current session. Use Undo after you have made a change you do not want to implement. Immediately after you select .Undo, the .Redo command becomes available, allowing you to restore what you just undid. You cannot undo the following operations: any change of view (e.g., Zoom-in or Zoom-out); any file operations (e.g., Open, Save, or Import); any selection operations (e.g., Marquee select or Node select).

Example

```
.Undo
```

The above example undoes the last command.

{button ,AL(`OVR1 Edit commands;',0,"Defaultoverview",)} Related Topics

View commands

RedrawAllScreens (DRAW)

.RedrawAllScreens

This command forces CorelDRAW to redraw all open document windows.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

RedrawScreen (DRAW)

.RedrawScreen

This command forces CorelDRAW to redraw the windows of the active document.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

ResumePainting (DRAW)

.ResumePainting

This command instructs CorelDRAW to resume screen updating. To stop screen updating, use the .SupressPainting command.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

SetFullScreenPreview (DRAW)

.SetFullScreenPreview .FullScreen = *boolean*

This command removes everything but your drawing from the screen. You cannot edit your drawing in this mode.

Parameter	Description
.FullScreen	Set to TRUE (-1) to remove everything but your drawing from the screen. Set to FALSE (0) to return to normal mode.

Example

```
.SetFullScreenPreview -1
```

The above example displays a full-screen preview of the active image.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

SetVisible (DRAW)

.SetVisible .Visible = *boolean*

This command makes the CorelDRAW Script Edtior visible.

Parameter	Description
.Visible	Set to TRUE (-1) to show the CorelDRAW Script Editor. Set to False (0) to hide the CorelDRAW Script Editor.

Example

```
.SetVisible -1
```

The above example makes the CorelDRAW Script Editor visible.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

SuppressPainting (DRAW)

.SuppressPainting ShowDialog = *boolean*

This command instructs CorelDRAW to suppress screen updating. To resume screen updating, use the .ResumePainting command.

Parameter	Description
.ShowDialog	Set to TRUE (-1) displays the script running dialog box. Set to FALSE (0) disables the script running message dialog box.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

ZoomIn (DRAW)

.ZoomIn

This command changes the view by zooming in twice the size.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

ZoomOut (DRAW)

.ZoomOut

This command changes the view by zooming out half the size.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

ZoomToAllObjects (DRAW)

.ZoomToAllObjects

This command changes the view by zooming to all the objects on the page.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

ZoomToHeight (DRAW)

.ZoomToHeight

This command changes the view by zooming in to the page height.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} Related Topics

ZoomToPage (DRAW)

.ZoomToPage

This command changes the view by zooming to the page.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

ZoomToRectangle (DRAW)

.ZoomToRectangle .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*

This command changes the view by zooming in on a specified area.

Parameter	Description
.Top	Lets you specify the Y coordinate of the top of the rectangle in tenths of a micron.
.Left	Lets you specify the X coordinate of the left of the rectangle in tenths of a micron.
.Bottom	Lets you specify the Y coordinate of the bottom of the rectangle in tenths of a micron.
.Right	Lets you specify the X coordinate of the right of the rectangle in tenths of a micron.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

ZoomToSelection (DRAW)

.ZoomToSelection

This command changes the view by zooming in on the current selection.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

ZoomToWidth (DRAW)

.ZoomToWidth

This command changes the view by zooming in on the page width.

{button ,AL(`OVR1 View commands;',0,"Defaultoverview",)} **Related Topics**

Layout commands

AddPageFrame (DRAW)

.AddPageFrame

This command puts a printable background frame around the page.

Example

```
.AddPageFrame
```

The above example creates a frame around the new page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

ChangeLayerColor (DRAW)

.ChangeLayerColor .LayerName = *string,* .PageNum = *long*

This command lets you set the outline color for the override color..

Parameter	Description
.LayerName	Lets you specify the name of the Layer.
.PageNum	Lets you specify the pafe number: 0 = Master

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

ChangeLayerName (DRAW)

.ChangeLayerName .NewLayerName = *string*

This command lets you assign a new name to the active layer.

Parameter	Description
.NewLayerName	Lets you specify the new name of the Layer.

Example

```
.ChangeLayerName "NewName"
```

The above example changes the layer name to "NewName."

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

CopyToLayer (DRAW)

.CopyToLayer .LayerName = *string*

This command places a copy of the selected object on the layer indicated in the LayerName.

Parameter	Description
.LayerName	Lets you specify the name of the destination layer.

Example

```
.CreateRectangle -200000, 200000, -900000, 900000, 0
.CopyToLayer "Layer2"
```

The above example creates a rectangle and copies it to "Layer2."

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

DeleteLayer (DRAW)

.DeleteLayer .LayerName = *string*

This command deletes the active layer and any objects on it.

Parameter	Description
.LayerName	Lets you specify the name of the layer to delete. This parameter is optional. If no layer name is specified the active layer is deleted.

Example

```
.MoveToLayer "NewLayer1"
.DeleteLayer
```

The above example moves to the layer named "NewLayer 1" and deletes it.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

DeletePages (DRAW)

.DeletePages .UnusedParameter, .NumberOfPages = *long, .*StartPage = *long*

This command deletes pages from the current drawing.

Parameter	Description
.UnusedParameter	This parameter is not used
.NumberOfPages	Lets you specify the number of pages to delete. Note: The current page is included in the deletion.
.StartPage	Lets you specify the page number to begin deleting pages to delete.

Example

```
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 4
.PasteFromClipboard
.DeletePages , 2, 2
```

The above example inserts 4 pages after the current page, pastes the contents of the Clipboard on the fourth page, then deletes two pages starting on the second page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

DisplayFacingPages (DRAW)

.DisplayFacingPages .FacingPages = *boolean*, .LeftFirst = *boolean*

This command displays two consecutive pages on the screen at the same time.

Parameter	Description
.FacingPages	Set to TRUE (-1) to display two consecutive pages on the screen at the same time. Working in this view allows you to draw objects that lie partially on both pages at the same time. Set to FALSE (0) to disable this option.
.LeftFirst	Set to TRUE (-1) to display odd pages on the left. Set to FALSE (0) to display odd pages on the right.

Example

```
.FileNew
.DisplayFacingPages 0, -1 'Displays one page
```

The above example displays one page.

```
.FileNew
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CreateRectangle 750000, -750000, 0, 0, 0
.CopyToClipboard
.InsertPages 0, 4
.PasteFromClipboard
.DisplayFacingPages -1, -1 'Displays two pages
```

The above example displays facing pages with the current page on the left.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

EditLayer (DRAW)

.EditLayer .LayerName = *string*, .PageNum = *long,* .NewName = *string,* .Visible = *boolean,* .Printable = *boolean,* .Locked = *boolean,* .Master = *boolean,* .WireOverride = *boolean,* .ChangeColor = *boolean*

This command sets all the layer properties.

Parameter	Description
.LayerName	Lets you specify the layer name.
.PageNum	Lets you specify the page number.
,NewName	Lets you specify the new name for the layer.
.Visible	Set to TRUE (-1) makes the layer visible.
.Printable	Set to TRUE (-1) makes the layer printable.
.Locked	Set to TRUE (-1) makes the layer locked.
.Master	Set to TRUE (-1) makes the layer the master layer.
.WireOverride	Set to TRUE (-1) forces the view to wireframe display.
.ChangeColor	Set to TRUE (-1) changes the color of the wireframe view.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

GetCurrentPageName (DRAW)

.GetCurrentPageName

This command returns the name of the current page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

GetCurrentPageOrientation (DRAW)

.GetCurrentPageOrientation .Orient = *long*

This command returns the orientation of the current document page.

Parameter	Description
.Orient	Lets you specify the page orientation: 0 = Portrait 1 = Landscape

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

GetCurrentPageSize (DRAW)

.GetCurrentPageSize .Width = *long,* .Height = *long*

This command returns the width and height of the current document page.

Parameter	Description
.Width	Returns the width of the current page in tenths of a micron.
.Height	Returns the height of the current page in tenths of a micron.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

GetDocumentCount (DRAW)

.GetDocumentCount

This command returns the number of open documents.

Example

lCount& = .GetDocumentCount()

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

GetDocumentName (DRAW)

.GetDocumentName .Name = s*tring**

This command returns the name of the current document.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

GetPageCount (DRAW)

.GetPageCount

This command returns the number of pages in the current document.

Parameter	Description
.Name	Returns the name of the current document.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

GetPageSize (DRAW)

.GetPageSize .Width = *long**, .Height = *long**

This command returns the width and height of the document page.

Parameter	Description
.Width	Returns the width of the page in tenths of a micron.
.Height	Returns the height of the page in tenths of a micron.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

InsertPages (DRAW)

.InsertPages .BeforePage = *boolean*, .NumberOfPages = *long*[*,* .StartPageNumber = *long*]

This command inserts the specified number of pages into the current drawing.

Parameter	Description
.BeforeCurrentPage	Set to TRUE (-1) to position insertion point before the current page. Set to FALSE (0) to position insertion point after the current page.
.NumberOfPages	Lets you specify the number of pages to insert.
.StartPageNumber (optional)	Lets you specify the page number from where to insert pages.

Example

```
.InsertPages 0, 4, 5
```

The above example inserts 4 pages after the current page, starting on the fifth page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

MoveLayerTo (DRAW)

.MoveLayerTo .DestLayer = *string,* .BeforeDest = *boolean*

This command moves the current layer to another position.

Parameter	Description
.DestLayer	Lets you specify the name of the destination layer.
.BeforeDest	Set to TRUE (-1) places the layer before the destination layer.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

MoveToLayer (DRAW)

.MoveToLayer .LayerName = *string*

This command moves the selected object to the layer selected in the Layers list.

Parameter	Description
.LayerName	Lets you specify the name of the destination layer.

Example

```
.MoveToLayer "NewLayer1"
```

The above example moves the selected object(s) to the layer named "NewLayer1."

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

NewLayer (DRAW)

.NewLayer .LayerName = *string*

This command lets you create a new layer and assign a name.

Parameter	Description
.LayerName	Lets you specify the name of the new layer.

Example

```
.NewLayer "NewLayer1"
```

The above example creates a new layer named "NewLayer1."

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SelectLayer (DRAW)

.SelectLayer .LayerName = *string*

This command lets you select a layer, making it the active layer.

Parameter	Description
.LayerName	Lets you specify the name of the selected layer.

Example

```
.SelectLayer "NewLayer1"
```

The above example selects the layer named "NewLayer1" and makes it the active layer.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetApplyToDuplicate (DRAW)

.SetApplyToDuplicate .ApplyToDuplicate = *boolean*

This command opens and closes a block of object-duplicating commands. An object must be selected to use this command. The duplicated object can be repositioned, resized, skewed, or rotated.

Parameter	Description
.ApplyToDuplicate	Set to TRUE (-1) to open a block of object-duplicating commands. Set to FALSE (0) to close the block.

Note

- The following commands can be used to duplicate objects within the .SetApplyToDuplicate block:

  .SetPosition
  .SkewObject
  .SetSize
  .RotateObject

  The duplicated object is selected.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.SetPosition 55555, 900000
.SetApplyToDuplicate TRUE
.SetPosition 0, 0 'Creates another object
.ApplyUniformFillColor 2, 255, 0, 0, 0
.SetPosition 55555, 100000 'Creates another object
.ApplyUniformFillColor 2, 0, 255, 0, 0
.SkewObject -15000000, 2000000, 3 'Creates another object
.SetSize 444444, 555555 'Creates another object
.RotateObject 45000000, 0, 0, 0 'Creates another object
.SetApplyToDuplicate FALSE
.SetPosition 0, 0
```

The above example creates an ellipse then creates 5 more ellipses in the SetApplyToDuplicate block.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetColorOverride (DRAW)

.SetColorOverride .Override = *boolean*

This command outlines objects on a layer in the selected color. Objects on the selected layer will appear with a wireframe outline of the chosen color.

Parameter	Description
.Override	Set to TRUE (-1) to outline objects on a layer in the selected color. Set to FALSE (0) to disable this option.

Example

```
.StoreColor DRAW_COLORMODEL_PANTONE, 3, 255, 0, 0
.SetColorOverride
```

The above example sets the override color to cyan.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetCurrentDocument (DRAW)

.SetCurrentDocument .

This command makes the selected document the current document.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetCurrentPage (DRAW)

.SetCurrentPage .CurrentPage = *long*

This command makes the specified page the current page.

Parameter	Description
.CurrentPage	Lets you specify which page to make the current page.

Example

```
.SetCurrentPage 2
```

The above example sets the second page as the current page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetCurrentPageName (DRAW)

.SetCurrentPageName .Name = *string*

This command sets the current page name.

Parameter	Description
.Name	Lets you specify the name for the current page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetCurrentPageOrientation (DRAW)

.SetCurrentPageOrientation .Orient = *long*

This command sets the orientation for the current page.

Parameter	Description
.Orient	Lets you specify the page orientation: 0 = Portrait 1 = Landscape

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetCurrentPageSize (DRAW)

.SetCurrentPageSize .Width = *long,* .Height = *long*

This command sets the size for the current page.

Parameter	Description
.Width	Lets you specify the page width in tenths of a micron.
.Height	Lets you specify the page height in tenths of a micron.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetDocVisible (DRAW)

.SetDocVisible .Show = *boolean*

This command makes the current document visible or hidden.

Parameter	Description
.Show	Set to TRUE (-1) to make a document visible. Set to FALSE (0) to hide a document.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetLayerLocked (DRAW)

.SetLayerLocked .Locked = *boolean*[, .LayerName = *string*][, .PageNum = *long*]

This command enables or disables selection of objects on a layer. Locking a layer prevents objects on it from being accidentally moved or changed in any way. You cannot add new objects to a locked layer.

Parameter	Description
.Locked	Set to TRUE (-1) to lock a layer, preventing objects on it from being accidentally moved or changed in any way. You cannot add new objects to a locked layer. Set to FALSE (0) to unlock a layer.
.LayerName (optional)	Lets you specify the layer name.
.PageNum (optional)	Specifes the page number. 0=Master

Example

```
.SetLayerLocked -1, 3, 2
```

The above example locks third layer on the second page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetLayerPrintable (DRAW)

.SetLayerPrintable .Printable = *boolean*[, .LayerName = *string*][, .PageNum = *long*]

This command enables or disables printing of objects on the current layer.

Parameter	Description
.Printable	Set to TRUE (-1) to enable printing of the current layer. Set to FALSE (0) to disable printing of the current layer.
.LayerName (optional)	Lets you specify the layer name.
.PageNum (optional)	Specifes the page number. 0=Master

Note

- If .SetOptionsForAllPages is set TRUE (-1), then the .SetLayerPrintable command applies to all pages.

Example

```
.SetLayerPrintable 0, 3, 2
```

The above example disables printing of the third layer on the second page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetLayerVisible (DRAW)

.SetLayerVisible .Visible = *boolean*[, .LayerName = *string*][, .PageNum = *long*]

This command makes objects on a layer visible or invisible.

Parameter	Description
.Visible	Set to TRUE (-1) to make the current layer visible. Set to FALSE (0) to make the current layer invisible.
.LayerName (optional)	Lets you specify the layer name.
.PageNum (optional)	Specifes the page number. 0=Master

Note

- If .SetOptionsForAllPages is set TRUE (-1), then the .SetLayerVisible command applies to all pages.

Example

```
.SetLayerVisible -1, 3, 2
```

The above example makes the third layer on the second page visible.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetMultiLayer (DRAW)

.SetMultiLayer .MultiLayer = *boolean*

This command lets you select objects on all layers that are not locked or invisible.

Parameter	Description
.MultiLayer	Set to TRUE (-1) to enable selection of objects across all layers except those which are locked or invisible. Set to FALSE (0) to disable selection of objects across all layers—only objects on the current layer can be selected.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetOptionsForAllPages (DRAW)

.SetOptionsForAllPages .AllPages = *boolean*

This command enables CorelDRAW options to be set for all pages.

Parameter	Description
.AllPages	Set to TRUE (-1) to enable options to be set for all pages. Set to FALSE (0) to disable this option.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetPageLayout (DRAW)

.SetPageLayout .LayoutType = *long*

This command lets you specify a page layout.

Parameter	Description
.LayoutType	Lets you specify the style of the page layout: 1 = Full Page: Prints one full page per sheet. 2 = Book: Prints two pages per sheet, which you would cut down the middle. 3 = Booklet: Prints two pages per sheet, which you would fold vertically to obtain a side fold. 4 = Tent Card: Prints two pages per sheet, which you would fold horizontally to obtain a top fold. 5 = Side-Fold Card: Prints four pages per sheet, which you would fold first horizontally to create the top fold, then vertically to create the side fold. 6 = Top-Fold Card: Prints four pages per sheet, which you would fold first vertically to create the side fold, then horizontally to create the top fold.

Example

```
.SetPageLayout 3
```

The above example sets the page layout to booklet style.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetPageOrientation (DRAW)

.SetPageOrientation .Orient = *long*

This command changes the orientation of the page.

Parameter	Description
.Orient	DRAW_ORIENT_PORTRAIT = portrait DRAW_ORIENT_LANDSCAPE = landscape

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetPageOrientation command.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetPageSize (DRAW)

.SetPageSize .Width = *long*, .Height = *long*

This command lets you set the page size for the document.

Parameter	Description
.Width	Lets you specify the new page width in tenths of a micron.
.Height	Lets you specify the new page height in tenths of a micron.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.SetPageSize 1000000,1350000
```

The above example sets the page size to 1,000,000 microns wide by 1,350,000 microns high (or 3.94 inches by 5.31 inches).

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetPageSizeFromPrinter (DRAW)

.SetPageSizeFromPrinter

This command sets the page size and orientation of the current document to the current settings of the default printer.

Example

```
.SetPageSizeFromPrinter
```

The above example queries the printer to set the page size.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetPaperColor (DRAW)

.SetPaperColor .

This command lets you color the Preview screen (and the Drawing Window, if you are working in the Editable Preview) to approximate the paper you plan to print it on.

Example

```
.StoreColor DRAW_COLORMODEL_CMYK, 0, 255, 0, 0
.SetPaperColor
```

The above example sets the paper color to magenta.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} **Related Topics**

SetPosition (DRAW)

.SetPosition .XPos = *long*, .YPos = *long*

This command sets the position for placement of the selected object

Parameter	Description
.XPos	Lets you specify the X-coordinate of the new position in tenths of a micron.
.YPos	Lets you specify the Y-coordinate of the new position in tenths of a micron.

Example

```
.CreateRectangle 1350000, -1000000, 750000, -500000, 0
.CreateArtisticText "1"
.SetPosition -950000, 1250000
```

The above example creates a rectangle and positions a number '1' in its upper-left corner.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetReferencePoint (DRAW)

.SetReferencePoint .ReferencePoint = *long*

This command sets the specified Reference Point for a selected object. The reference point is used to set the object handle for subsequent commands such as .SetPosition.

Parameter	Description
.ReferencePoint	Lets you specify the reference point to set. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetReferencePoint command.

Example

```
.CreateRectangle 1250000, -1000000, 750000, -500000, 0
.SetReferencePoint 9
.SetPosition 0, 0
```

The above example creates a rectangle, sets its reference point to the center and positions it in the center of the page.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetSize (DRAW)

.SetSize .XSize = *long*, .YSize = *long*

This command lets you scale, mirror, or set the size of the selected object.

Parameter	Description
.XSize	Lets you specify the new horizontal size of the selected object, in tenths of a micron.
.YSize	Lets you specify the new vertical size of the selected object, in tenths of a micron.

Note

- To mirror an object, use negative values for the .XSize and .YSize parameters.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
id& = .GetObjectsCDRStaticID()
status& = .GetSize (XSize&, YSize&)
.SelectObjectOfCDRStaticID id&
.SetSize 2*XSize&, 3*YSize&
```

The above example gets the size of the selected rectangle and sets the width to twice the original size, and the height to three times the original size.

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
id& = .GetObjectsCDRStaticID()
status& = .GetSize (XSize&, YSize&)
.SelectObjectOfCDRStaticID id&
.SetSize -XSize&, YSize&
```

The above example horizontally mirrors the selected object, maintaining its original size.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

SetToMasterLayer (DRAW)

.SetToMasterLayer .Master = *boolean*[, .LayerName = *string*]

This command lets you set the selected object to a master layer. When you want the same element, for example, a company logo, to appear on every page of a document, use this command to set the "master layers" to contain the repeating elements.

Parameter	Description
.Master	Set to TRUE (-1) to enable, applying the Master Layer template to all layers. Set to FALSE (0) to disable this option.
.LayerName (optional)	Lets you specify the layer name.

Example

```
.CreateRectangle 1350000, -1000000, 750000, -500000, 0
.SetToMasterLayer -1
```

The above example sets the rectangle to the master layer.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

ShowPageBorder (DRAW)

.ShowPageBorder .ShowBorder = *boolean*

This command enables and disables the page border.

Parameter	Description
.ShowBorder	Set to TRUE (-1) to show the page border. Set to FALSE (0) to suppress the page border.

Example

```
.ShowPageBorder -1
```

The above example shows the page border.

```
.ShowPageBorder 0
```

The above example hides the page border.

{button ,AL(`OVR1 Layout commands;',0,"Defaultoverview",)} Related Topics

Styles commands

ApplyStyle (DRAW)

.ApplyStyle .Style = *string*

This command lets you apply a style to the selected object.

Parameter	Description
.Style	Lets you specify the name of the style.

Example

```
.SelectAllObjects
.ApplyStyle "Default Graphic"
```

The above example applies the 'Default Graphic' style to all selected objects.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} Related Topics

CreateNewStyle (DRAW)

.CreateNewStyle .StyleType = *long,* .StyleName = *string*

This command lets you create a new style.

Parameter	Description
.StyleType	Lets you specify the type of style: 0 = Artistic text 1 = Paragraph text 2= Graphic
.StyleName	Lets you specify the name of the new style.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} Related Topics

DeleteStyle (DRAW)

.DeleteStyle .Style = *string*

This command deletes styles. When you delete a style, objects with that style revert to the default style for that object type. The object's appearance does not change when it reverts to the default style.

Parameter	Description
.Style	Lets you specify the name of the style to delete.

Example

```
.DeleteStyle "Style 1"
```

The above example deletes the style named "Style 1."

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} **Related Topics**

LoadStyles (DRAW)

.LoadStyles .StyleSheet = *string*

This command loads the styles from a template into the active drawing.

Parameter	Description
.StyleSheet	Lets you specify the name of the template to use.

Example

```
.LoadStyles "C:\mine.cdt"
```

The above example loads the styles from the template file "MINE.CDT" into the active document.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} Related Topics

RenameStyle (DRAW)

.RenameStyle .OldName = *string*, .NewName = *string*

This command renames a style name.

Parameter	Description
.OldName	Lets you specify the old name of the style.
.NewName	Lets you specify the new name of the style.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} **Related Topics**

RevertToStyle (DRAW)

.RevertToStyle

This command converts an object to its original style.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} **Related Topics**

SaveStyleAs (DRAW)

.SaveStyleAs .Style = *string*, .Fill = *boolean*, .Outline = *boolean*, .Typeface = *boolean*, .TypeStyle = *boolean*, .Size = *boolean*, .Justification = *boolean*, .Tabs = *boolean*, .Hyphenation = *boolean*, .SpaceChar = *boolean*, .SpaceWord = *boolean*, .SpaceLine = *boolean*, .BeforePara = *boolean*, .AfterPara = *boolean*, .Underline = *boolean*, .Overline = *boolean*, .Strikeout = *boolean*, .BulletIndent = *boolean*, .FirstLineIndent = *boolean*, .RestOfLinesIndent = *boolean*, .RightMargin = *boolean*, .SuperOrSubScript = *boolean*, .Capitalize = *boolean*, .Bullet = *boolean*

This command saves the style of the current object as a new style.

Parameter	Description
.Style	Lets you specify the name of the new style.
.Fill	Set to TRUE (-1) to include fill properties. Set to FALSE (0) to exclude these properties.
.Outline	Set to TRUE (-1) to include outline properties. Set to FALSE (0) to exclude these properties.
.Typeface	Set to TRUE (-1) to include typeface properties. Set to FALSE (0) to exclude these properties.
.TypeStyle	Set to TRUE (-1) to include type style properties. Set to FALSE (0) to exclude these properties.
.Size	Set to TRUE (-1) to include size properties. Set to FALSE (0) to exclude these properties.
.Justification	Set to TRUE (-1) to include text justification properties. Set to FALSE (0) to exclude these properties.
.Tabs	Set to TRUE (-1) to include tab stop properties. Set to FALSE (0) to exclude these properties.
.Hyphenation	Set to TRUE (-1) to include hyphenation properties. Set to FALSE (0) to exclude these properties.
.SpaceChar	Set to TRUE (-1) to include character spacing properties. Set to FALSE (0) to exclude these properties.
.SpaceWord	Set to TRUE (-1) to include word spacing properties. Set to FALSE (0) to exclude these properties.
.SpaceLine	Set to TRUE (-1) to include line spacing properties. Set to FALSE (0) to exclude these properties.
.BeforePara	Set to TRUE (-1) to include paragraph spacing properties (before the paragraph). Set to FALSE (0) to exclude these properties.
.AfterPara	Set to TRUE (-1) to include paragraph spacing properties (after the paragraph). Set to FALSE (0) to exclude these properties.
.Underline	Set to TRUE (-1) to include text underline properties. Set to FALSE (0) to exclude these properties.
.Overline	Set to TRUE (-1) to include text overline properties. Set to FALSE (0) to exclude these properties.
.Strikeout	Set to TRUE (-1) to include text strikeout properties. Set to FALSE (0) to exclude these properties.
.BulletIndent	Set to TRUE (-1) to include bullet indentation properties. Set to FALSE (0) to exclude these properties.
.FirstLineIndent	Set to TRUE (-1) to include indentation properties for the first line. Set to FALSE (0) to exclude these properties.
.RestOfLinesIndent	Set to TRUE (-1) to include indentation properties for remaining lines (hanging indent). Set to FALSE (0) to exclude these properties.
.RightMargin	Set to TRUE (-1) to include right margin properties. Set to FALSE (0) to exclude these properties.
.SuperOrSubScript	Set to TRUE (-1) to include superscript or subscript properties. Set to FALSE (0) to exclude these properties.
.Capitalize	Set to TRUE (-1) to include capitalization properties. Set to FALSE (0) to exclude these properties.
.Bullet	Set to TRUE (-1) to include bullet properties. Set to FALSE (0) to exclude these properties.

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} Related Topics

SaveStyleProp (DRAW)

.SaveStyleProp .StyleName = *string,* .UseFill = *boolean,* .UseOutline = *boolean,* .UseFont = *long,* .UseAlignment = *long,* .UseSpacing = *long,* .UseLines = *long,* .UseIndentsAndMargins = *long,* .UseTextEffects = *long*

This command lets you save the style based on the current properties.

Parameter	Description
.StyleName	Lets you specify the name of the new Style.
.UseFill	Set to TRUE (-1) saves the fill as part of the style.
.UseOutline	Set to TRUE (-1) saves the outline as part of the style.
.UseFont	Lets you specify the font: 1 = Typeface 2 = Typestyle 4 = Size
.UseAlignment	Specifices the alignement: 1 = Justification 2 = Tabs 4 = Hyphenation
.UseSpacing	Lets you specify the spacing: 1 = InterChar Spacing 2 = InterWord Spacing 4 = Interline Spacing 8 = Spacing Before Paragraphs
.UseLines	Lets you specify the lines: 1 = Underline 2 = Overline 4 = Strikeout
.UseIndentsAndMargins	Lets you specify the indents and margins: 1 = Bullet Indent 2 = First Line Indent 4 = Rest of Lines Indent 8 = Right Margin
.UseTextEffects	Lets you specify the text effect 1 = Superscript/Subscript 2 = Capitalization Effect 4= Bullet

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} **Related Topics**

SaveTemplate (DRAW)

.SaveTemplate .StyleSheet = *string*

This command lets you save the styles in the active document as a template.

Parameter	Description
.StyleSheet	Lets you specify the name of the Style Sheet to save.

Example

```
.SaveTemplate "C:\TMPLATE1.CDT"
```

The above example saves a template named "TMPLATE1.CDT".

{button ,AL(`OVR1 Styles commands;',0,"Defaultoverview",)} Related Topics

Object selection commands

AfterObject (DRAW)

.AfterObject .ObjectID = *long*

This command selects the object that is after the reference object in the object tree. Use .lObjectID to specify the reference object.

Parameter	Description
.ObjectID	Lets you specify the object ID of the reference object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

AppendObjectToSelection (DRAW)

.AppendObjectToSelection .ObjectID = *long*

This command adds the object with the specified object ID to the existing selection.

Parameter	Description
.ObjectID	Lets you specify the object ID of the object to append. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

BeforeObject (DRAW)

.BeforeObject .ObjectID = *long*

This command selects the object that is before the reference object in the object tree. Use .lObjectID to specify the reference object.

Parameter	Description
.ObjectID	Lets you specify the object ID of the reference object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

ClearNodeSelection (DRAW)

.ClearNodeSelection

This command clears the selection of nodes.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

FindNextObjectOfStyle (DRAW)

ReturnValue = .FindNextObjectOfStyle()

This function finds the next object with the current style.

Return Value

Returns one of the following values

- TRUE (-1) an object is found
- FALSE (0) no object is found.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

FindObjectOfStyle (DRAW)

ReturnValue = .FindObjectOfStyle(.StyleName = *string*)

This function finds the next object with the specified style.

Return Value

Returns one of the following values:

- TRUE (-1)▸an object is found
- FALSE (0)▸no object is found.

Parameter	Description
.StyleName	Lets you specify the name of the style.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

GetObjectType (DRAW)

ReturnValue& = .GetObjectType()

This function returns a value that indicates the type of selected object. If more than one object is selected, the function returns the type of the last selected object.

Return Value

Returns one of the following values:

- 0 Reserved for future use
- 1 Rectangle
- 2 Ellipse
- 3 Curve
- 4 Text
- 5 Bitmap
- 6 Paragraph Text
- 7 OLE
- 9 Symmetrical Polygon
- 12 Grouped objects

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .GetObjectType command.

Example

```
objType& = .GetObjectType()
MESSAGE objType&
```

The above example displays a number that corresponds to the type of selected object in a message box.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

GetUserClick (DRAW)

.GetUserClick .XCoord = *long*, .YCoord = *long*,[.Timeout = *long*,][.IgnoreSnap = *boolean*]

This function returns the coordinates of a user click.

Parameter	Description
.XCoord	Returns the X coordinate of the selection click.
.YCoord	Returns the Y coordinate of the selection click.
.Timeout (optional)	Lets you specify the amount of time, in seconds, to wait for the user to click. Defaults to 10.
.IgnoreSnap (optional)	Defaults to TRUE (-1) 0 does not ignore the snap -1 ignores the snap

---

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

IsSelection (DRAW)

.IsSelection

This command returns true if an object is selected.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

SelectAllObjects (DRAW)

.SelectAllObjects

This command selects every object in your drawing, including any not currently in view.

Example

```
.SelectAllObjects
```

The above example selects all objects in the active document.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

SelectNextNode (DRAW)

.SelectNextNode

This command selects the next node in the current curve.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

SelectNextObject (DRAW)

.SelectNextObject .SelectInsideGroup = *boolean*

This command lets you select the next object in the drawing. Repeat this command until the object you want is selected.

Parameter	Description
.SelectInsideGroup	Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE (0) to disable this option.

Example

```
.SelectNextObject -1
```

The above example selects the next object in the drawing. If that object is in a group, it can be selected.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

SelectNode (DRAW)

.SelectNode .Index = *long,* .AddToSelection = *boolean*

This command lets you select the I-th node of the current curve.

Parameter	Description
.Index	Lets you specify the I-th node of the current curve.
.AddToSelection	Set to TRUE (-1) adds the specified node to the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

SelectNodeAt (DRAW)

.SelectNodeAt .X = *long,* .Y = *long,* .AddToSelection = *boolean*

This command lets you select a specific node.

Parameter	Description
.X	Lets you specify the X coordinate of the node in the current curve.
.Y	Lets you specify the Y coordinate of the node in the current curve.
.AddToSelection	Set to TRUE (-1) adds the specified node to the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

SelectObjectAtPoint (DRAW)

.SelectObjectAtPoint(.XPos = *long*, .YPos = *long*, SelectInsideGroup = *boolean*)

This command toggles the selection of an object at the specified point. Using this command is the same as holding down SHIFT and clicking an object during a DRAW session.

Parameter	Description
.XPos	Lets you specify one of the X-coordinates of the selected object in tenths of a micron, relative to the center of the page.
.YPos	Lets you specify one of the Y-coordinates of the selected object in tenths of a micron, relative to the center of the page.
.SelectInsideGroup	Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE to disable this option.

Example

```
.CreateRectangle 1350000, -1000000, 1300000, 0, 0
.CreateRectangle 1000000, -750000, 500000, 100000, 0
.CreateRectangle 100000, -500000, -100000, 50000, 0
.CreateRectangle -750000, -500000, -250000, 50000, 0
.UnSelectAll
.SelectObjectAtPoint -750000, 500000, 0
.ApplyUniformFillColor 2, 255, 0, 0, 0
```

The above example creates four rectangles, then selects the second one and fills it with cyan.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

SelectObjectOfCDRStaticID (DRAW)

.SelectObjectOfCDRStaticID .CDRStaticID = *long*

This command selects the object with the specified CDRStaticID.

Parameter	Description
.CDRStaticID	Lets you specify the CDRStaticID number of the object to select.

Example

```
.CreateRectangle 750000, -600000, 250000, -100000, 0
IDRect& = .GetObjectsCDRStaticID()
.SelectObjectOfCDRStaticID IDRect&
```

The above example demonstrates object selection using the object's CDRStaticID.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

SelectObjectOfType (DRAW)

.SelectObjectOfType .Type = *long*

This command selects the object of a given type.

Parameter	Description
.Type	Lets you specify the type of object to select.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

SelectObjectsInRect (DRAW)

.SelectObjectsInRect .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .IncludeIntersecting = *boolean*

This command selects all objects found within the defined rectangular area

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the X-coordinate of the lower-right corner of the distribution rectangle in tenths of a micron, relative to the center of the page.
.IncludeIntersecting	Set to TRUE (-1) to included intersecting objects in the selection. Set to FALSE (0) to disable this option.

Example

```
.SelectObjectsInRect 1350000, -1000000, -1350000, 1000000, 0
```

The above example selects all objects within the specified rectangle.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

SelectPreviousObject (DRAW)

.SelectPreviousObject .SelectInsideGroup = *boolean*

This command lets you select the previously selected object in the drawing. Repeat this command until the object you want is selected. The objects are selected in the order in which they were created.

Parameter	Description
.SelectInsideGroup	Set to TRUE (-1) to permit object selection within a group of objects. Set to FALSE (0) to disable this option.

Example

```
.SelectPreviousObject -1
```

The above example selects the previous object in the group.

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} **Related Topics**

UnSelectAll (DRAW)

.UnSelectAll

This command deselects all objects.

Example

```
.UnSelectAll
```

The above example deselects all selected object(s).

{button ,AL(`OVR1 Object selection commands;',0,"Defaultoverview",)} Related Topics

Object creation commands

AddBezierPoint (DRAW)

.AddBezierPoint .X = *long*, .Y = *long*, .Constrain = *boolean*, .Cusp = *boolean*

This command creates the second point of a bezier segment created by using the .BeginDrawBezier command. The segment itself is not added until the .EndDrawBezier command is called.

Parameter	Description
.X	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
.Constrain	Set to TRUE (-1) to use the constrain angle when positioning the point.
.Cusp	Set to TRUE (-1) to make the new node cusped. Set to FALSE (0) to make it symmetrical (except for end nodes).

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

AddFreehandPoint (DRAW)

.AddFreehandPoint .ConvertToDPCoords = *boolean*, .X = *long*, .Y = *long*

This command adds a point to a freehand curve created by using the .BeginDrawFreehand command. The segment itself is not added until .EndDrawFreehand command is called.

Parameter	Description
.ConvertToDPCoords	Set to TRUE (-1) to convert the X and Y coordinates to physical coordinates on the screen. This parameter should always be set to true unless you know that the coordinates you are using are already physical coodinates on the screen.
.X	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

AppendCurveLine (DRAW)

.AppendCurveLine .X1 = *long*, .Y1 = *long*, .X2 = *long*, .Y2 = *long*

This command adds a line segment to an existing curve. An open curve must be selected.

Parameter	Description
.X1	Lets you specify the X-coordinate of the start point in tenths of a micron, relative to the center of the page.
.Y1	Lets you specify the Y-coordinate of the start point in tenths of a micron, relative to the center of the page.
.X2	Lets you specify the X-coordinate of the end point in tenths of a micron, relative to the center of the page.
.Y2	Lets you specify the Y-coordinate of the end point in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

BeginDrawBezier (DRAW)

.BeginDrawBezier .X = *long*, .Y = *long*, .Cusp = *boolean*

This command creates the first point of a bezier segment. The second point is added by the .AddBezierPoint command. The segment itself is not added until the .EndDrawBezier command is called. This command must be preceeded by the .InitBezierTool command.

Parameter	Description
.X	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
.Cusp	Set to TRUE (-1) to make the new node cusped. Set to FALSE (0) to make it symmetrical (except for end nodes).

Note

- The bezier commands include:

  .AddBezierPoint
  .MoveBezierControl

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

BeginDrawCurve (DRAW)

.BeginDrawCurve .X = *long*, .Y = *long*

This command sets the coordinates of the starting node when drawing curves in Freehand mode.

Parameter	Description
.X	Lets you specify the X-coordinate of the starting node of the curve in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the starting node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .BeginDrawCurve command must be followed by a contiguous block of one or more DrawCurve commands, and one .EndDrawCurve command. The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

BeginDrawFreehand (DRAW)

.BeginDrawFreehand .ConvertToDPCoords = *boolean*, .X = *long*, .Y = *long*

This commandcreates the first point of a freehand segment. Additional points are added to the segment by using the .AddFreehandPoint command. The curve itself is not added until the .EndDrawFreehand command is called.

Parameter	Description
.ConvertToDPCoords	Set to TRUE (-1) to convert the X and Y coordinates to physical coordinates on the screen. This parameter should always be set to true unless you know that the coordinates you are using are already physical coodinates on the screen.
.X	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

CloneObject (DRAW)

.CloneObject .XOffset = *long*, .YOffset = *long*

This command copies the selected object and offsets the copy from the original. Most changes applied to the original object (called the "master") are automatically applied to the copy (called the "clone"). For example, if you change the master's fill, the clone's fill will change as well. If you change the attributes of the clone, the attribute you change will no longer depend on the master's attributes. For example, after you change a clone's fill, its fill will no longer change when you change the master's fill. Likewise, if you stretch a clone, it will no longer stretch when you stretch its master.

Parameter	Description
.XOffset	Lets you specify the horizontal distance to offset the clone object.
.YOffset	Lets you specify the vertical distance to offset the clone object.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CloneObject
```

The above example creates an ellipse, then makes a clone.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

ConvertToCurves (DRAW)

.ConvertToCurves

This command converts the selected polygon, rectangle, ellipse, or text object to a series of curves you can shape with the Shape tool.

Example

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.ConvertToCurves
```

The above example converts the selected rectangle to a curve object.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

CreateAngleDimension (DRAW)

.CreateAngleDimension .X1 = *long*, Y1= *long*, X2 = *long*, Y2 = *long*, X3 = *long*, Y3 = *long*, X4 = *long*, Y4 = *long*, LargeAngle = *boolean,* WitnessExtension = *long*, WitnessGap = *long*, LabelGap = *long*

This command is used to create an angular dimension.

Parameter	Description
.1X1, 1Y1	Lets you specify the apex coordinates.
.1X2, 1Y2	Lets you specify the baseline coordinates.
.1X3, 1Y3	Lets you specify the endline coordinates.
.1X4, 1Y4	Lets you specify the text point.
.LargeAngle	Set to TRUE the angle measured is <= 180 degrees. Set to FALSE the angle measured is >180 degrees.
.WitnessExtension	Lets you specify the length of the extension line that protrudes beyond the dimension line.
.WitnessGap	Lets you specify the distance between the snapped to object and the extension line.
.LabelGap	Lets you specify the amount of space between the text and the dimension line.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateCallout (DRAW)

.CreateCallout .X1 = *long*, Y1 = *long*, X2 = *long*, Y2 = *long*, X3 = *long*, Y3 = *long*, Text = *string*

This command is used to create a callout line that points to and labels an object.

Parameter	Description
.X1, Y1	Lets you specify where the first callout segment starts.
.X2, Y2	Lets you specify where the first callout segment ends.
.X3, Y3	Lets you specify the location of the callout text.
.Text	Lets you specify the callout text.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateConnector (DRAW)

.CreateConnector .X1 = *long*, Y1 = *long*, X2 = *long*, Y2 = *long*, Placement = *boolean*

This command connects objects with a line. When you move an object that has a connector line attached the connetor line also moves.

Parameter	Description
.X1, Y1	Lets you specify the first coordinate of the line.
.X2, Y2	Lets you specify the second coordinate of the line.
.Placement	Set to TRUE to keep connector line fixed to the nodes that it was originally attached to. Set to FALSE to draw the shortest line between the two objects it connects.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateDimension (DRAW)

.CreateDimension .X1 = *long*, Y1 = *long*, X2 = *long*, Y2 = *long*, X3 = *long*, Y3 = *long*, Style = *long*, WitnessExtention = *long*, WitnessGap = *long*, LabelGap = *long*

This command creates a horizontal, vertical or slanted dimension line.

Parameter	Description
.X1, Y1	Lets you specify the start point coordinate of the dimension line.
.X2, Y2	Lets you specify the end point coordinate of the dimension line.
.X3, Y3	Lets you specify the dimension text location.
.Style	0=vertical dimension
	1=horizontal dimension
	2=slanted dimension
.WitnessExtention	Lets you specify the length of the extension line that protrudes beyond the dimension line.
.WitnessGap	Lets you specify the distance between the snapped to object and the extension line.
.LabelGap	Lets you specify the amount of space between the text and the dimension line.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateEllipse (DRAW)

.CreateEllipse .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .StartAngle = *long*, .EndAngle = *long*, .Arc = *boolean*

This command draws ellipses and circles.

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the X-coordinate of the lower-right corner of the bounding rectangle of the ellipse in tenths of a micron, relative to the center of the page.
.StartAngle	If .CreateEllipse is used to create an arc, .StartAngle specifies the starting angle in degrees.
.EndAngle	If .CreateEllipse is used to create an arc, .EndAngle specifies the end angle, in degrees.
.Arc	Lets you specify whether to draw the ellipse as a pie or an arc. Set to TRUE (-1) to turn the ellipse into a pie. Set to FALSE (0) to draw the ellipse as an arc.

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
```

The above example creates an ellipse.

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000*
count), -500000+( 200000* count), 0, 0, 0
next count
```

The above example creates 4 ellipses.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateGridBoxes (DRAW)

.CreateGridBoxes .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .Wide = *long*, .High = *long*

This command draws a grid box.

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the X-coordinate of the lower-right corner of the bounding rectangle of the grid box in tenths of a micron, relative to the center of the page.
.Wide	Lets you specify the number of cells wide.
.High	Lets you specify the number of cells high.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateRectangle (DRAW)

.CreateRectangle .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*[, .CornerRadius = *long*][, .CornerRadius2 = *long*][, .CornerRadius3 = *long*][, .CornerRadius4 = *long*]

This command draws rectangles and squares.

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the rectangle in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the rectangle in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the rectangle in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the X-coordinate of the lower-right corner of the rectangle in tenths of a micron, relative to the center of the page.
.CornerRadius (optional)	Lets you specify the radius used to create the rounded corners in tenths of a micron.
.CornerRadius2 (optional)	Lets you specify the radius used to create the rounded corners in tenths of a micron.
.CornerRadius3 (optional)	Lets you specify the radius used to create the rounded corners in tenths of a micron.
.CornerRadius4 (optional)	Lets you specify the radius used to create the rounded corners in tenths of a micron.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
```

The above example creates a rectangle.

```
FOR count% = 1 TO 8
.CreateRectangle 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000*
count), -500000+( 200000* count), 0
NEXT count
```

The above example creates 8 rectangles.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateSpiral (DRAW)

.CreateSpiral .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .NumRevolutions = *long*, .SpiralType = *long,* .GrowthRate = *long*

This command draws a spiral.

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the bounding rectangle of the spiral in tenths of a micron, relative to the center of the page.
.NumRevolutions	Lets you specify the number of revolutions in the spiral.
.SpiralType	Lets you specify the type of spiral. 0 = Symmetrical 1 = Logarithmic
.GrowthRate	Lets you specify the growth rate of the spiral.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

CreateSymPolygon (DRAW)

.CreateSymPolygon .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .Sides = *long*, .Subpaths = *long*, .Complexity = *long*, .Star = *boolean*, .StarComplexity = *long*, .MaxComplexity = *long*

This command creates a polygon or star.

Parameter	Description
.Top	Lets you specify the coordinate of the top of the shape in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the coordinate of the left of the shape in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the coordinate of the bottom of the shape in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the coordinate of the right of the shape in tenths of a micron, relative to the center of the page.
.Sides	Lets you specify the number of sides (from 3 to 500).
.Subpaths	Lets you specify the number of subpaths in a polygon. If both the number of subpaths and the complexity are set to 1 then the polygon will be a simple polygon. If either of these values are greater than 1 then the polygon becomes a star. The relationship between the number of subpaths and the complexity is represented by the Star/Polygon button and the Sharpness slider on the Polygon Property Bar. Appropriate values for each of these parameters change depending on the number of sides of the polygon.
.Complexity	Lets you specify the complexity of a polygon. If both the number of subpaths and the complexity are set to 1 then the polygon will be a simple polygon. If either of these values are greater than 1 then the polygon becomes a star. The relationship between the number of subpaths and the complexity is represented by the Star/Polygon button and the Sharpness slider on the Polygon Property Bar. Appropriate values for each of these parameters change depending on the number of sides of the polygon. If there is only 1 subpath and the complexity is set to 2, then you will always create a simple star. For more complex stars, we recommend you experiment with the recorder to determine the appropriate values.
.Star	Set to TRUE (-1) to enable the StarComplexity parameter. Set to FALSE (0) to ignore this parameter. This parameter must be TRUE if you want to create a star-shaped polygon.
.StarComplexity	Lets you specify the distance that the start node is from the center of the shape. If this parameter is set to 0, the start node will remain at the outer edge of the shape. The closer this parameter's value is to the .lMaxComplexity value, the closer the start node is to the center of the shape. This parameter is equivalent to the Sharpness slider for polygons (as opposed to stars).
.MaxComplexity	Lets you specify the maximum value for star complexity.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

DeleteObject (DRAW)

.DeleteObject

This command deletes selected objects.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.CreateRectangle 750000, -750000, 0, 0, 0
.DeleteObject
```

The above example deletes the selected object. Since the rectangle is the last object created, it is selected and gets deleted.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DistributeObjects (DRAW)

.DistributeObjects .HorizontalDistribution = *long*, .VerticalDistribution = *long*, .ObjectOrPageExtents = *long*

This command distributes selected objects.

Parameter	Description
.HorizontalDistribution	Lets you specify the type of horizontal distribution. 0 = None 1 = Right edges of object 2 = Left edges of object 3 = Center edges of object 4 = Space between objects
.VerticalDistribution	Lets you specify the type of vertical distribution. 0 = None 1 = Top edges of object 2 = Bottom edges of object 3 = Center edges of object 4 = Space between objects
.ObjectOrPageExtents	Lets you specify the type of distribution. 0 = Extent of Selection 1 = Extent of Page

Example

```
.SelectAllObjects
.DistributeObjects 3, 3, 1
```

The above example distributes the selected objects to the center of the page.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DrawCurveClosePath (DRAW)

.DrawCurveClosePath

This command closes the path on the last node when drawing curves in Freehand mode.

Note

- The .DrawCurveClosePath command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveCurveTo 500000, 500000, 1000000 ,-500000, -500000, -500000
.DrawCurveClosePath
.EndDrawCurve
```

The above example draws an object in the shape of an uppercase "D".

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DrawCurveCurveTo (DRAW)

.DrawCurveCurveTo .X1 = *long*, .Y1 = *long*, .X2 = *long*, .Y2 = *long*, .XEnd = *long*, .YEnd = *long*

This command sets a node in a curve drawn in Freehand mode.

Parameter	Description
.X1	Lets you specify the X-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that was created in a preceding BeginDrawCurve or DrawCurveCurveTo command.
.Y1	Lets you specify the Y-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that was created in a preceding BeginDrawCurve or DrawCurveCurveTo command.
.X2	Lets you specify the X-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that is specified in the current DrawCurveCurveTo command.
.Y2	Lets you specify the Y-coordinate for a control point in tenths of a micron, relative to the center of the page. The control point is used with the node that is specified in the current DrawCurveCurveTo command.
.XEnd	Lets you specify the X-coordinate of a node of the curve in tenths of a micron, relative to the center of the page.
.YEnd	Lets you specify the X-coordinate of a node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveCurveTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveCurveTo 500000, 500000, 1000000 ,-500000, -500000, -500000
.DrawCurveCurveTo 600000, 600000, 1100000 ,-600000, -600000, -600000
.EndDrawCurve
```

The above example draws a curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DrawCurveLineTo (DRAW)

.DrawCurveLineTo .X = *long*, .Y = *long*

This command sets the coordinates when drawing continuous curves in Freehand mode.

Parameter	Description
.X	Lets you specify the X-coordinate of the next node of the curve in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the next node of the curve in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveLineTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DrawCurveMoveTo (DRAW)

.DrawCurveMoveTo .X = *long*, .Y = *long*

This command sets the coordinates when drawing non-continuous curves in Freehand mode.

Parameter	Description
.X	Lets you specify the X-coordinate of the point to move to without drawing in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the point to move to without drawing in tenths of a micron, relative to the center of the page.

Note

- The .DrawCurveMoveTo command must be in a contiguous block of one or more DrawCurve commands. The first DrawCurve command in the block must be preceded by the .BeginDrawCurve command, and the last must be followed by the .EndDrawCurve command. The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.DrawCurveMoveTo -500000, -500000
.DrawCurveLineTo 500000, 1000000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

DuplicateObject (DRAW)

.DuplicateObject .XOffset = *long*, .YOffset = *long*

This command adds a copy of the selected object(s) to the current drawing. By default, the copy is placed on top of the original, offset up and to the right. It is also selected automatically.

Parameter	Description
.XOffset	Lets you specify the horizontal distance to offset the duplicate object.
.YOffset	Lets you specify the vertical distance to offset the duplicate object.

Example

```
.CreateRectangle 1082025, -333882, 272052, 500823, 0, 0, 0, 0
.StoreColor 5002, 100, 0, 100, 0, 0, 0, 0, 0
.ApplyUniformFillColor
.DuplicateObject
```

The above example creates a rectangle and fills it with a color, then duplicates the object.

Note

- The created object is created on top of the object you duplicated.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

EndDrawBezier (DRAW)

.EndDrawBezier

This command ends a set of bezier creation commands that began with the .BeginDrawBezier command.

Note

- The bezier commands include:

  .AddBezierPoint
  .MoveBezierControl

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

EndDrawCurve (DRAW)

.EndDrawCurve

This command ends a set of curve creation commands that began with the .BeginDrawCurve command.

Note

- The DrawCurve commands include:

  .DrawCurveClosePath
  .DrawCurveCurveTo
  .DrawCurveLineTo
  .DrawCurveMoveTo

Example

```
.BeginDrawCurve -500000, 1000000
.DrawCurveLineTo 500000, -500000
.EndDrawCurve
```

The above example demonstrates the DrawCurve commands.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

EndDrawFreehand (DRAW)

.EndDrawFreehand .StraightTightness = *long*, .CornerTightness = *long*, .CornerThreshold = *long*, .SnapTightness = *long*

This command ends a set of frehand drawing commands that began with the .BeginDrawCurve command.

Parameter	Description
.StraightTightness	Sets the amount the curve can vary from a straight path and still be treated as straight. The higher the value, the less accurate the line needs to be. The valid range is from 1 to 10 pixels.
.CurveTightness	Determines how closely the curve will match the position of each point. The lower the number, the more accurate the match. The valid range is from 1 to 10 pixels.
.CornerThreshold	Sets the limit at which each corner node is cusped (as opposed to smooth). A node is more likely to be cusped if the value is lower. The valid range is from 1 to 10 pixels.
.SnapTightness	Determines how close two end nodes must be to join automatically. The valid range is from 1 to 10 pixels.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

GetEllipseClockwise (DRAW)

.GetEllipseClockwise

This function returns false of the ellipse is counterclockwise.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetEllipseEndAngle (DRAW)

.GetEllipseEndAngle

This function returns the end angle of the ellipse.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

GetEllipseStartAngle (DRAW)

.GetEllipseStartAngle

This function returns the start angle of the ellipse.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

GetEllipseType (DRAW)

.GetEllipseType

This function returns the type of ellipse. (0 = Ellipse, 1 = Arc, 2 = Pie)

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetObjectCount (DRAW)

.GetObjectCount .Selection = *boolean*, .Grouped = *boolean*

This function counts the number of objects.

Parameter	Description
.Selection	Set to TRUE (-1) counts objects in the current selection.
.Grouped	Set to TRUE (-1) counts groups as one object.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetObjectID (DRAW)

.GetObjectID .Index = *long*, .Selection = *boolean,* .Grouped = *boolean*

This function returns the ID of the Index-th object.

Parameter	Description
.Index	Lets you specify the object ID. 0 ▸ GetObjectCount-1
.Selection	Set to TRUE (-1) counts objects in the current selection.
.Grouped	Set to TRUE (-1) counts groups as one object.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

GetObjectsCDRStaticID (DRAW)

ReturnValue& = .GetObjectsCDRStaticID()

This function returns the CDRStaticID of the selected object. If more than one object is selected, the function returns the CDRStaticID of the last selected object.

Return Value

Returns the following value:

- the CDRStaticID of the selected object

Note

- Every object you create has a unique CDRStaticID in a document.

Example

```
.CreateRectangle 750000, -600000, 250000, -100000, 0
IDRect& = .GetObjectsCDRStaticID()
.SelectObjectOfCDRStaticID IDRect&
```

The above example demonstrates object selection using the object's CDRStaticID.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetPolygonSharpness (DRAW)

.GetPolygonSharpness

This function returns the sharpness value in a polygon.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetPolygonSides (DRAW)

.GetPolygonSides

This function returns the number of sides in a polygon.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

GetPolygonType (DRAW)

.GetPolygonType

This function returns the type of polygon. (0 = Polyong, 1 = Star, 2 = Polygon as star)

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

GetRectangleRadius (DRAW)

.GetRectangleRadius .Radius1 = *double*, .Radius2 = *double*, .Radius3 = *double*, .Radius 4 = *double*

This function returns the radius for each of a rectangle's corners.

Parameter	Description
.Radius1	Lets you specify the upper left radius of the rectangle.
.Radius2	Lets you specify the upper right radius of the rectangle.
.Radius3	Lets you specify the lower right radius of the rectangle.
.Radius4	Lets you specify the lower right radius of the rectangle.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

InitBezierTool (DRAW)

.InitBezierTool

This command must preceed the .BeginDrawBezier command.

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} Related Topics

MoveBezierControl (DRAW)

.MoveBezierControl .X = *long*, .Y = *long*, .Constrain = *boolean*

This command controls the position of bezier node control points created with the .BeginDrawBezier and .AddBezierPoint commands.

Parameter	Description
.X	Lets you specify the X-coordinate of the control point's position in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of the control point's position in tenths of a micron, relative to the center of the page.
.Constrain	Set to TRUE (-1) to use the constrain angle when positioning the control points.

Example

```
.InitBezierTool
.BeginDrawBezier -3085992, 163280, FALSE
.MoveBezierControl -1649128, 1142960, FALSE
.AddBezierPoint 146952, -277576, FALSE, FALSE
.MoveBezierControl -326560, -1453192, FALSE
.EndDrawBezier
```

The above example draws a simple bezier curve.

{button ,AL(`OVR1 Object creation commands;',0,"Defaultoverview",)} **Related Topics**

Node editing commands

AddNode (DRAW)

.AddNode .X = *long* , .Y = *long*

This command adds a node to the current curve at a specified location.

Parameter	Description
.X	Lets you specify the X coordinate.
.Y	Lets you specify the Y coordinate.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

BeginEditObject (DRAW)

.BeginEditObject

This command initializes a block of node editing commands that includes one or more instances of the .EditObjectCommand and ends with the .EndEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

CloseCurve (DRAW)

.CloseCurve

This command closes the selected open path.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

DeleteNode (DRAW)

.DeleteNode

This command deletes the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

EditAngleDimensionLabel (DRAW)

.EditAngleDimensionLabel .Dynamic = *boolean*, Precision = *short*, Units = *short*, ShowUnits = *short*, Prefix = *string*, Suffix = *string*

This command lets you change the properties of an angle dimension label.

Parameter	Description
.Dynamic	Set to TRUE the dimension text changes automatically when the objects is stretched or skewed.
.Precision	0=0
	1=0.0
	2=0.00
	3=0.000
	4=0.0000
	5=0.00000
	6=0.000000
	7=0.0000000
	8=0.00000000
	9=0.000000000
	10=0.0000000000
.Units	0=degrees
	1=radians
	2=gradians
.ShowUnits	Set to TRUE displays the units beside the dimension text.
.Prefix	Lets you specify that the prefix is attached to the dimension text.
.Suffix	Lets you specify that the suffix is attached to the dimension text.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

EditDimensionLabel (DRAW)

.EditDimensionLabel .Placement = *short*, Horizontal = *boolean*, Center = *boolean*, Dynamic = *boolean*, Style = *short*, Precision = *short*, Units = *short*, ShowUnits = *boolean*, Prefix = *string*, Suffix = *string*

This command lets you change the properties of a vertical, horizontal or slanted dimension label.

Parameter	Description
.Placement	0=Places text above the dimension line.
	1=Places text within the dimension line.
	2=Places text below the dimension line.
.Horizontal	Set to TRUE places the text horizontally.
.Center	Set to TRUE centers the text in the dimension line.
.Dynamic	Set to TRUE the dimension text changes automatically when the objects is stretched or skewed.
.Style	0=Decimal
	1=Fractional
	2=U.S. Engineering
	3=U.S. Architectural
.Precision	0=0 or 0
	1=0.0 or 01/2 or 0'-0" or 0'-0" depending on .sStyle
	2=0.00 or 01/4 or 0'-0.0" or 0'-01/4" depending on .sStyle
	3=0.000 or 01/8 or 0'-0.00" or 0'-01/8" depending on .sStyle
	4=0.0000 or 01/16 or 0'-0.000" or 0'-01/16" depending on .sStyle
	5=0.00000 or 01/32 or 0'-0.0000" or 0'-01/32" depending on .sStyle
	6=0.000000 or 01/64 or 0'-0.00000" or 0'-01/64" depending on .sStyle
	7=0.0000000 or 01/128 or 0'-0.000000" or 0'-01/128" depending on .sStyle
	8=0.00000000 or 01/256 or 0'-0.0000000" or 0'-01/256" depending on .sStyle
	9=0.000000000 or 01/512 or 0'-0.00000000" or 0'-01/512" depending on .sStyle
	10=0.0000000000 or 01/1024 or 0'-0.000000000" or 0'-01/1024" depending on .sStyle
.Units	0="
	1=in
	2=inches
	3='
	4=ft
	5=mi
	6=miles
	7=yds
	8=yards
	9=m
	10=meters
	11=km
	12=kilometers
	13=cm
	14=centimeters
	15=mm
	16=millimeters
	17=picas
	18=points
	19=ciceros
	20=didots
.ShowUnits	Set to TRUE displays the units beside the dimension text.

.Prefix	Lets you specify that the prefix is attached to the dimension text.
.Suffix	Lets you specify that the suffix is attached to the dimension text.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

EditObjectCommand (DRAW)

.EditObjectCommand .Cmd = *long*, .X = *long*, .Y = *long*, .Key = *long*, .AddToSelection = *boolean*

This command lets you change the shape of objects by editing their nodes. This command must be part of a block of commands that begins with the .BeginEditObject command and ends with the .EndEditObject command.

Parameter	Description
.Cmd	0 = This value is equivalent to releasing the mouse button. Use with the .lX parameter and the .lY parameter. 1 = This value is equivalent to pressing one of the nudge or super-nudge keys. Use the .lKey parameter to indicate which nudge key is used. 2 = This value is equivalent to pressing a key on the keyboard. Use the .lKey parameter to indicate which key is used. 3 = This value is equivalent to pressing down the mouse button. Use with the .lX parameter and the .lY parameter. 4 = This value is equivalent to the add node option. 5 = This value is equivalent to the delete node option. 6 = This value is equivalent to the join nodes option. 7 = This value is equivalent to the break path option. 8 = This value makes a node cusped. 9 = This value makes a node smooth. 10 = This value makes a segment straight. 11 = This value makes a segment curved. 12 = This value makes a node symmetrical. 13 = This value is equivalent to the auto-reduce nodes option. 14 = This value is equivalent to the extract subpath option. 16 = This value toggles the elastic option.
.X	Lets you specify the X-coordinate of a node in tenths of a micron, relative to the center of the page.
.Y	Lets you specify the Y-coordinate of a node in tenths of a micron, relative to the center of the page.
.Key	If .lCmd = 1 0 = nudge left 1 = nudge right 2 = nudge up 3 = nudge down 4 = super-nudge left 5 = super-nudge right 6 = super-nudge up 7 = super-nudge down If .lCmd = 2 0 = HOME 1 = END 2 = TAB 3 = ESC
.AddToSelection	Set to TRUE (-1) to select multiple nodes (equivalent to pressing SHIFT).

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

EndEditObject (DRAW)

.EndEditObject

This command ends a block of node editing commands that includes one or more instances of the .EditObjectCommand and begins with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

GetCurveClose (DRAW)

.GetCurveClose

This command returns true if the curve is closed.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveFirstNodePosition (DRAW)

.GetCurveFirstNodePosition .X = *long* , .Y = *long*

This command returns the coordinates of the first node in the selected curve.

Parameter	Description
.X	Lets you specify the X coordinate.
.Y	Lets you specify the Y coordinate.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveIthNodePosition (DRAW)

.GetCurveIthNodePosition .X = *long* , .Y = *long*

This command returns the coordinates of the I-th node in the selected curve.

Parameter	Description
.X	Lets you specify the I-th coordinate.
.Y	Lets you specify the I-th coordinate.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveLastNodePosition (DRAW)

.GetCurveLastNodePosition .X = *long* , .Y = *long*

This command returns the coordinates of the last node in the selected curve.

Parameter	Description
.X	Lets you specify the X coordinate.
.Y	Lets you specify the Y coordinate.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveLength (DRAW)

This command returns the curve length.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveNodeCount (DRAW)

This command returns the number of nodes of the selected curve.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetCurveSubpathCount (DRAW)

This command returns the number of subpaths of the selected curve.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetNodeIndex (DRAW)

.GetNodeIndex .Position= *long*

This command returns the index of the current node.

Parameter	Description
.Position	Returns the position index of the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

GetNodePosition (DRAW)

.GetNodePosition .X= *long,* .Y = *long,* .Position= *long*

This command returns the index of the current node.

Parameter	Description
.X	Lets you specify the X coordinate of the current node.
.Y	Lets you specify the Y coordinate of the current node.
.Position	Lets you specify the position of the current node.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetNodeSelectedCount (DRAW)

This command returns the number of nodes in the current selection.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

GetNodeType (DRAW)

Return Value = **.GetNodeType**

This command returns the type of node in the current selection.

Return Value

Returns of the following values:

- 0 Cusp
- 1 Smooth
- 2 Symmetrical

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

GetSegmentLength (DRAW)

.GetSegmentLength

This command returns the segment length for the current selection.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

GetSegmentType (DRAW)

Return Value = .GetSegmentType

This command returns the segment type for the current selection.

Return Value

Returns one of the following values:

- 0 Line
- 1 Curve

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

MoveNode (DRAW)

.MoveNode .DeltaX = *long,* .DeltaY = *long*

This command moves a node a specifies distance.

Parameter	Description
.DeltaX	Lets you specify the amount to move the node horizontally.
.DeltaY	Lets you specify the amount to move the node vertically.

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

SetCornerRoundness (DRAW)

.SetCornerRoundess .Roundness = *long* , .Roundness2 = *long,* .Roundness3 = *long,* .Roundness4 = *long*

This command sets the corner roundness of the selected object.

Parameter	Description
.Roundness	Lets you specify the roundness of the corners in tenths of a percent.
.Roundness2	Lets you specify the roundness of the corners in tenths of a percent.
.Roundness3	Lets you specify the roundness of the corners in tenths of a percent.
.Roundness4	Lets you specify the roundness of the corners in tenths of a percent.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

SetEllipseProperties (DRAW)

.SetEllipseProperties .Arc = *boolean*, .StartAngle = *long*, .EndAngle = *long*

This command sets the ellipse properties.

Parameter	Description
.Arc	Set to TRUE = Arc. Set to FALSE = Ellipse/Pie
.StartAngle	Lets you specify the start angle in millionths of a degree.
.EndAngle	Lets you specify the end angle in millionths of a degree.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

SetNodeType (DRAW)

.SetNodeType .Type = *long*

This command sets the type of node.

Parameter	Description
.Type	Lets you specify the type of node: 0 = Cusp 1 = Smooth 2 = Symmetrical

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

SetPolygonProperties (DRAW)

.SetPolygonProperties .Star = *boolean,* .Points = *long,* .Sharpness = *long*

This command sets the polygon properties of the selected object.

Parameter	Description
.Star	If set to TRUE = Star. If set to FALSE = Polygon.
.Points	Lets you specify the number of points/edges in a polygon.
.Sharpness	Speficies the sharpness of the star.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} **Related Topics**

SetSegmentType (DRAW)

.SetSegmentType .Type = *long*

This command sets the type of segment.

Parameter	Description
.Type	Lets you specify the type of node: 0 = Line 1 = Curve

Note

- You must precede this command with the .BeginEditObject command.

{button ,AL(`OVR1 Node editing commands;',0,"Defaultoverview",)} Related Topics

Symbols commands

DropSymbol (DRAW)

.DropSymbol .SymbolLibrary = *string*, .SymbolNumber = *long*, .Tile = *boolean*, .XPosOrGridSize = *long*, .YPosOrGridSize = *long*, .ProportionalSizing = *boolean*, .SymbolSize = *long*

This command positions the specified symbol at the defined position or the specified grid position.

Parameter	Description
.SymbolLibrary	Lets you specify the name of the Symbol Library. Refer to the Symbols dialog box for more details.
.SymbolNumber	Lets you specify the Symbol Index Number, which identifies the selected symbol. Refer to the Symbols dialog box for more details.
.Tile	Set to TRUE (-1) to create a pattern from the selected symbol that fills the page. Set to FALSE (0) to disable this option. Note that the tiled symbols are clones of the upper left symbol.
.XPosOrGridSize	Lets you specify the X-coordinate or grid position at which to place the symbol, in tenths of a micron.
.YPosOrGridSize	Lets you specify the Y-coordinate or grid position at which to place the symbol, in tenths of a micron.
.ProportionalSizing	Set to TRUE (-1) to enable proportional sizing of the symbol. Set to FALSE (0) to disable this option.
.SymbolSize	Lets you specify the size of the symbol in tenths of a micron. The symbol can be resized after it's been added to your drawing.

Example

```
.DropSymbol "Animals 1", 42, 0, 0, 0, 0, 1000000
```

The above example places a kangaroo symbol in the center of the page.

Arrange commands

AlignObjects (DRAW)

.AlignObjects .HorizontalAlignment = *long*, .VerticalAlignment = *long*

This command aligns the selected objects according to the last selected object.

Parameter	Description
.HorizontalAlignment	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
.VerticalAlignment	Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignObjects 2, 0
```

The above example aligns the selected objects according to the last selected object.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

AlignToCenterOfPage (DRAW)

.AlignToCenterOfPage .HorizontalAlignment = *long*, .VerticalAlignment = *long*

This command aligns selected objects to the center of the page.

Parameter	Description
.HorizontalAlignment	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
.VerticalAlignment	Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignToCenterOfPage 0, 3
```

The above example vertically aligns all objects to the center of the page.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

AlignToGrid (DRAW)

.AlignToGrid .HorizontalAlignment = *long*, .VerticalAlignment = *long*

This command aligns the selected objects to the gridpoint nearest to the edge of the selection.

Parameter	Description
.HorizontalAlignment	Lets you specify the type of horizontal alignment. 0 = None 1 = Right 2 = Left 3 = Center
.VerticalAlignment	Lets you specify the type of vertical alignment. 0 = None 1 = Top 2 = Bottom 3 = Center

Example

```
.SelectAllObjects
.AlignToGrid 1, 0
```

The above example horizontally aligns all objects to a gridpoint, nearest to the left edge of the selection.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

BreakApart (DRAW)

.BreakApart

This command converts an object made up of multiple subpaths into individual curve objects.

Example

```
.BreakApart
```

The above example breaks apart the selected object into individual curve objects.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

Combine (DRAW)

.Combine

This command combines the selected curve or line segments into a single object. If you use Combine on rectangles, ellipses, polygons, or text, CorelDRAW converts them to curves before converting them into a single curve object. However, when text is combined with other text it is not converted to curves; it is converted to larger blocks of text.

Example

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000*
count), -500000+( 200000* count), 0, 0, 0
next count
.SelectAllObjects
.Combine
.ApplyUniformFillColor 2, 0, 255, 0, 0
```

The above example creates 4 ellipses, then combines them before applying a fill.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

ConvertOutlineToObject (DRAW)

.ConvertOutlineToObject

This command converts an object's outline into an object.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

CreateGuidelineUsingAngle (DRAW)

.CreateGuidelineUsingAngle .IntersectPointX = *long*, .IntersectPointY = *long*, .Angle = *long*, .Locked = *boolean*

This command creates a guideline at a specific location using an intersection point and an angle to specify where to put the new guideline.

Parameter	Description
.IntersectPointX	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.IntersectPointY	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
.Angle	Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).
.Locked	Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

CreateGuidelineUsingTwoPoints (DRAW)

.CreateGuidelineUsingTwoPoints .Point1X = *long*, .Point1Y = *long*, .Point2X = *long*, .Point2Y = *long*, .Locked = *boolean*

This command creates a guideline at a specific location using two sets of coordinates to place the guidleine.

Parameter	Description
.Point1X	Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point1Y	Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point2X	Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Point2Y	Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Locked	Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

DeleteGuidelineByIndex (DRAW)

.DeleteGuidelineByIndex .Index = *long*

This command deletes a specific guideline.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

DeleteGuidelineUsingAngle (DRAW)

.DeleteGuidelineUsingAngle .IntersectX = *long*, .IntersectY = *long*, .Angle = *long*

This command deletes a specific guideline based on a set of coordinates and an angle.

Parameter	Description
.IntersectPointX	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.IntersectPointY	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
.Angle	Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

DeleteGuidelineUsingTwoPoints (DRAW)

.DeleteGuidelineUsingTwoPoints .Point1X = *long*, .Point1Y = *long*, .Point2X = *long*, .Point2Y = *long*

This command deletes a specific guideline using two sets of coordinates.

Parameter	Description
.Point1X	Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point1Y	Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point2X	Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Point2Y	Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

GetGuidelineInformation (DRAW)

.GetGuidelineInformation .Index = *long*, .Point1X = *long**, .Point1Y = *long**, .Point2X = *long**, .Point2Y = *long**, .InterceptX = *long**, .InterceptY = *long**, .Angle = *long**, .Locked = *boolean**

This command retrieves all the information available about a specific guideline. The .plPoint parameters are equivalent to the coordinates used in the .CreateGuidlineUsingTwoPoints command, and the .plIntercept and .plAngle parameters are equivalent to the parameters used in the .CreateGuidelineUsingAngle command.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
.Point1X	Returns the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point1Y	Returns the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point2X	Returns the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Point2Y	Returns the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
.InterceptX	Returns the X-coordinate of the point that relates to .plAngle in tenths of a micron, relative to the center of the page.
.InterceptY	Returns the Y-coordinate of the point that relates to .plAngle in tenths of a micron, relative to the center of the page.
.Angle	Returns the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).
.Locked	Returns TRUE (-1) if guidelines are locked. Returns FALSE (0) if the guidelines are unlocked.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

GetNumberOfGuidelines (DRAW)

ReturnValue& = .GetNumberOfGuidelines()

This function return the number of guidelines. You can use this function to determine the index of a guidline. The range of possible guideline index values is 0 to the number of guidelines minus 1.

Return Value

Returns the following value:

- the number of guidelines

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

Group (DRAW)

.Group

This command groups all selected objects together to allow them to be selected and manipulated as a single object.

Example

```
for count% = 1 to 4
.CreateEllipse 1500000-(250000 * count), -1200000 +( 200000* count), 750000 - ( 200000*
count), -500000+( 200000* count), 0, 0, 0
next count
.SelectAllObjects
.Group
.ApplyUniformFillColor 5, 0, 0, 255, 0
```

The above example groups the four ellipses together so that they are treated as one object, and applies a blue uniform fill to all four.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

Intersection (DRAW)

.Intersection .LeaveTarget = *Boolean*, .LeaveModifiers = *Boolean*

This command creates a new object using the area common to two or more overlapping objects. Intersection joins their paths at the points where they intersect. The resulting curve object assumes the fill and outline attributes of the last selected object.

Parameter	Description
.LeaveTarget	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
.LeaveModifiers	Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Intersection, -1, -1
```

The above example selects all objects and creates a new object(s) using the area common to overlapping objects. Both the target object and the modifier object are left in the document.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

LockGuidelineByIndex (DRAW)

.LockGuidelineByIndex .Index = *long*

This command locks a specific guideline.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

MoveGuidelineUsingAngleByIndex (DRAW)

.MoveGuidelineUsingAngleByIndex .Index = *long*, .InterceptX = *long*, .InterceptY = *long*, .Angle = *long*, .Locked = *boolean*

This command moves a specific guideline to a location using an intersection point and an angle to specify where to put the new guideline.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
.InterseptX	Lets you specify the X-coordinate of the point in tenths of a micron, relative to the center of the page.
.InterseptY	Lets you specify the Y-coordinate of the point in tenths of a micron, relative to the center of the page.
.Angle	Lets you specify the angle of the guideline in millionths of a degree (e.g., 5000000 = 5 degrees).
.Locked	Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

MoveGuidelineUsingTwoPointsByIndex (DRAW)

.MoveGuidelineUsingTwoPointsByIndex .Index = *long*, .Point1X = *long*, .Point1Y = *long*, .Point2X = *long*, .Point2Y = *long*, .Locked = *boolean*

This command moves a specific guideline to a location using two sets of coordinates to place the guidleine.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.
.Point1X	Lets you specify the X-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point1Y	Lets you specify the Y-coordinate of the first point in tenths of a micron, relative to the center of the page.
.Point2X	Lets you specify the X-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Point2Y	Lets you specify the Y-coordinate of the second point in tenths of a micron, relative to the center of the page.
.Locked	Set to TRUE (-1) to lock the guideline. Set to FALSE (0) to leave the guideline unlocked.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

OrderBackOne (DRAW)

.OrderBackOne

This command rearranges the stacking order by moving the selected object back one position.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.OrderBackOne
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The ellipse, still selected, is ordered back one position in the drawing.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

OrderForwardOne (DRAW)

.OrderForwardOne

This command rearranges the stacking order by moving the selected object up one position.

Example

```
.SelectObjectOfCDRStaticID Six&
.OrderForwardOne
```

The above example orders the selected object forward one position.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

OrderReverseOrder (DRAW)

.OrderReverseOrder

This command reverses the stacking order of the selected object(s).

Example

```
.SelectAllObjects
.OrderReverseOrder
```

The above example reverses the order of all the objects.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

OrderToBack (DRAW)

.OrderToBack

This command rearranges the stacking order by moving the selected object to the back of the screen. Areas of the object overlapped by other objects with fills are "knocked out" so that they will not print.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.OrderToBack
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The ellipse, still selected, is ordered to the back of the drawing.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

OrderToFront (DRAW)

.OrderToFront

This command rearranges the stacking order by moving the selected object to the front of the layer.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyUniformFillColor 5,255,0,0,0
.CreateEllipse -450000, -700000, 450000, 700000, 0, 0, 0
.ApplyUniformFillColor 5,0,0,250,0
.SelectPreviousObject 0
.OrderToFront
```

The above example creates a rectangle and then creates an ellipse on top of the rectangle. The rectangle is then selected and ordered to the front of the drawing.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

RemoveAllGuidelines (DRAW)

.RemoveAllGuidelines

This command removes all guidelines.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

Separate (DRAW)

.Separate

This command separates original objects from intermediate shapes.

Example

```
.Separate
```

The above example separates a combined object into its individual component object(s).

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

Trim (DRAW)

.Trim .LeaveTarget = *Boolean*, .LeaveModifiers = *Boolean*

This command lets you trim selected objects. Trimming two or more overlapping objects reshapes the last object selected. Trimming separates the paths at points where the objects overlap. Initially, the trimmed object may appear no different than it did before trimming. However, closer inspection will show that new nodes appear where the object was trimmed. Move the trimmed objects apart to see the full effect of the trim.

Parameter	Description
.LeaveTarget	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
.LeaveModifiers	Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Trim ,-1, -1
```

The above example trims the selected objects. Both the target object and the modifier object are left in the document.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

Ungroup (DRAW)

.Ungroup

This command breaks up the selected group into its individual objects. If you have more than one sublevel of grouping, Ungroup breaks up one level of grouping at a time.

Example

```
.Ungroup
```

The above example breaks up the grouped object into its individual object components.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

UngroupAll (DRAW)

.UngroupAll

This command breaks up all the groups into its individual objects.

Example

```
.UngroupAll
```

The above example breaks up all the grouped objects into its individual object components.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

UnlockGuidelineByIndex (DRAW)

.UnlockGuidelineByIndex .Index = *long*

This command unlocks a specific guideline.

Parameter	Description
.Index	Lets you specify the guidline ID. Use the .GetNumberOfGuidelines command to get a guideline's ID.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} Related Topics

Weld (DRAW)

.Weld . LeaveTarget = *Boolean*, LeaveModifiers = *Boolean*

This command joins overlapping objects at points where their paths intersect. Although not necessarily apparent in editable preview, welding also removes sections of the path between those intersect points. The resulting curve object assumes the fill and outline attributes of the bottom object of the selected group of objects. If you marquee-select the objects, CorelDRAW will outline and fill the welded object with the attributes of the most recently created object.

Parameter	Description
.LeaveTarget	Set to TRUE (-1) leaves the target object in your document. Set to FALSE (0) removes the target object from your document.
.LeaveModifiers	Set to TRUE (-1) leaves the modifier object in your document. Set to FALSE (0) removes the modifier object from your document.

Example

```
.SelectAllObjects
.Weld -1, -1
```

The above example welds the selected object group. Both the target object and the modifier object are left in the document.

{button ,AL(`OVR1 Arrange commands;',0,"Defaultoverview",)} **Related Topics**

Transformation commands

GetPosition (DRAW)

.GetPosition .XPos = *long**, .Ypos = *long**

This function returns the position coordinates of a selected object's reference point. If more than one object is selected, the function returns the position coordinates of the last selected object.

Parameter	Description
.XPos	Returns the X-coordinate of the selected object's reference point in tenths of a micron, relative to the center of the page. An object's default reference point is the upper-left corner.
.YPos	Returns the Y-coordinate of the selected object's reference point in tenths of a micron, relative to the center of the page. An object's default reference point is the upper-left corner.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100005, 0
id& = .GetObjectsCDRStaticID()
.GetPosition XPos&, YPos&
MESSAGE "Horizontal"+STR(XPos&)
MESSAGE "Vertical"+STR(YPos&)
```

The above example creates a rectangle then displays the coordinates of the upper-left corner in message boxes.

```
.CreateRectangle 1000000, 750000, 500000, 100005, 0
id& = .GetObjectsCDRStaticID()
.SetReferencePoint 3
.GetPosition XPos&, YPos&
MESSAGE "Horizontal"+STR(XPos&)
MESSAGE "Vertical"+STR(YPos&)
```

The above example creates a rectangle then displays the coordinates of the upper-right corner in message boxes. The upper-right coordinates are used because the selected object's reference point was changed with the .SetReferencePoint command.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

GetSize (DRAW)

.GetSize .XSize = *long**, .YSize = *long**

This function returns the size attributes of a selected object. If more than one object is selected, the function returns the size attributes of the last selected object.

Parameter	Description
.XSize	Returns the horizontal size of the selected object, in tenths of a micron.
.YSize	Returns the vertical size of the selected object, in tenths of a micron.

Example

```
.CreateRectangle 1000000, 750000, 450000, 100000, 0
id& = .GetObjectsCDRStaticID()
.GetSize XSize&, YSize&
MESSAGE "Horizontal"+STR(XSize&)
MESSAGE "Vertical"+STR(YSize&)
```

The above example returns the size of the selected rectangle and displays the width and height (in tenths of a micron) in message boxes.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

MoveCenter (DRAW)

.MoveCenter .AnchorID = *long*, .XOffset = *long*, .YOffset = *long*

This command moves the selected object's center of rotation.

Parameter	Description
.AnchorID	Lets you specify the reference point of the object to be skewed. -1 = Use .XOffset and YOffset 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center
.XOffset	If .lAnchorID = -1, then .XOffset specifies the distance, in tenths of a micron, that the center of rotation is horizontally offset from the object's center.
.YOffset	If .lAnchorID = -1, then .YOffset specifies the distance, in tenths of a micron, that the center of rotation is vertically offset from the object's center.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} **Related Topics**

MoveObject (DRAW)

.MoveObject .XDelta = *long*, .YDelta = *long*

This command repositions the selected object to the specified location.

Parameter	Description
.XDelta	Lets you specify the distance the object is to be moved along the X-axis in tenths of a micron.
.YDelta	Lets you specify the distance the object is to be moved along the Y-axis in tenths of a micron.

Example

```
.SetPageSize 2159000, 2794000
.CreateRectangle 500000, -750000, -500000, 750000, 0
.MoveObject 250000, -750000
```

The above example creates a rectangle, then moves it to the bottom right corner of an 8.5 by 11 inch page.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

ResetTransfo (DRAW)

.ResetTransfo

This command clears all tranformations.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

RotateObject (DRAW)

.RotateObject .Angle = *long*, .UseObjectsCenter = *boolean*, .XCenter = *long*, .YCenter = *long*

This command lets you rotate the selected object.

Parameter	Description
.Angle	Lets you specify the angle of rotation of the selected object, expressed in millionths of degrees. Negative values rotate the object clockwise from its current position; positive values rotate it counterclockwise. e.g., 45 degrees clockwise = -45000000
.UseObjectsCenter	Set to TRUE (-1) to enable rotation around the center of the object. Set to FALSE (0) to disable this option.
.XCenter	Lets you specify the logical X-coordinate of the center of the object to be rotated in tenths of a micron, relative to the center of the page.
.YCenter	Lets you specify the logical Y-coordinate of the center of the object to be rotated in tenths of a micron, relative to the center of the page.

Note

- You can use the ANGLECONVERT function to specify angle measurements.

Example

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.RotateObject 45000000, -1, 0,0
```

The above example rotates the rectangle 45 degrees counter clockwise.

```
.CreateRectangle 500000, -750000, -500000, 750000, 0
.RotateObject -45000000, 0, -500000, 500000
```

The above example rotates the rectangle 45 degrees clockwise about the specified point.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

SkewObject (DRAW)

.SkewObject .XAngle = *long*, .YAngle = *long*, .Reference = *long*

This command lets you skew the selected object.

Parameter	Description
.XAngle	Lets you specify the amount of horizontal skew (skew along the X-axis), in millionths of degrees. Positive angles result in counter-clockwise skew. Negative angles result in clockwise skew.
.YAngle	Lets you specify the amount of vertical skew (skew along the Y-axis), in millionths of degrees. Positive angles result in counter-clockwise skew. Negative angles result in clockwise skew.
.Reference	Lets you specify the reference point of the object to be skewed. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

Note

- You can use the ANGLECONVERT function to specify angle measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.SkewObject -15000000, 20000000, 3
```

The above example creates a rectangle, horizontally skews it 15 degrees clockwise and vertically skews it 20 degrees counterclockwise. The reference point for skewing is the upper-left position.

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} Related Topics

StretchObject (DRAW)

.StretchObject .XScaleNumerator = *long*, .XScaleDenominator = *long*, .YScaleNumerator = *long*, .YScaleDenominator = *long*, .bHMirror = *boolean*, .bVMirror = *boolean*, .ReferenceNum = *long*

This command stretches or mirrors the selected object.

Parameter	Description
.XScaleNumerator	Lets you specify the amount that the selected object is stretched along the X-axis. The final stretch value is determined by dividing this number by .XScaleDenominator. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
.XScaleDenominator	Lets you specify the amount that the selected object is stretched along the X-axis. The final stretch value is determined by dividing .XScaleNumerator by this number. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
.YScaleNumerator	Lets you specify the amount that the selected object is stretched along the Y-axis. The final stretch value is determined by dividing this number by .YScaleDenominator. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
.YScaleDenominator	Lets you specify the amount that the selected object is stretched along the Y-axis. The final stretch value is determined by dividing .YScaleNumerator by this number. If the final value is less than 1, the object becomes smaller. If the final value is greater than 1, the object becomes larger.
.HMirror	Set to TRUE (-1) to horizontally mirror the selected object.
.VMirror	Set to TRUE (-1) to vertically mirror the selected object.
.ReferenceNum	Lets you specify the reference point of the object to be skewed. 1 = Upper-right 2 = Upper-middle 3 = Upper-left 4 = Middle-left 5 = Lower-left 6 = Lower-middle 7 = Lower-right 8 = Middle-right 9 = Center

{button ,AL(`OVR1 Transformation commands;',0,"Defaultoverview",)} **Related Topics**

Text commands

AddTabStop (DRAW)

.AddTabStop .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .TabStop = *long*

This command adds tab stops to text.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.TabStop	Lets you specify the distance at which to apply tabs, in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.AddTabStop 0, 0, 1270000
```

The above example adds a tab stop every 0.5 inch.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

AlignTextToBaseline (DRAW)

.AlignTextToBaseline .FirstSelectedChar = *long*, .LastSelectedChar = *long*

This command aligns text to the baseline.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.AlignTextToBaseline 0, 0
```

The above example aligns the text to the baseline.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

CreateArtisticText (DRAW)

.CreateArtisticText .NewText = *string*, .Left = *long*, .Top = *long*

This command allows you to create a text string with default text settings as Artistic Text. The left-most character of the Artistic Text is placed on the center of the page. Text created with the CreateArtisticText command can be modified using the SetArtisticText command.

Parameter	Description
.NewText	Lets you specify the name of the new text to create.
.Left	Sets the left edge of the artistic text's position in tenths of a micron, relative to the center of the page.
.Top	Sets the top edge of the artistic text's position in tenths of a micron, relative to the center of the page.

Example

```
.CreateArtisticText "CorelDRAW", 1000, 1000
```

The above example displays the text string "CorelDRAW" 100 microns to the left of the center of the page, and 100 microns above the center of the page.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

CreateTextString (DRAW)

.CreateTextString .Top = *long*, .Left = *long*, .Bottom = *long*, .Right = *long*, .Text = *string*

This command creates the text.

Parameter	Description
.Top	Lets you specify the Y-coordinate of the upper-left corner of the text's bounding box in tenths of a micron, relative to the center of the page.
.Left	Lets you specify the X-coordinate of the upper-left corner of the text's bounding box in tenths of a micron, relative to the center of the page.
.Bottom	Lets you specify the Y-coordinate of the lower-right corner of the text's bounding box in tenths of a micron, relative to the center of the page.
.Right	Lets you specify the X-coordinate of the lower-right corner of the text's bounding box in tenths of a micron, relative to the center of the page.
.Text	Lets you specify the text. Maximum string length is 255 characters.

Note

- This function must be called first to create the text before any of the functions which manipulate the text.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
```

The above example creates the text "COREL".

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

ExtractText (DRAW)

.ExtractText .DestinationFile = *string*

This command extracts the Artistic Text to a text file which can then be edited in any text editor and merged back into the document with MergeTextBack.

Parameter	Description
.DestinationFile	Lets you specify the name of the destination file.

Example

```
.CreateArtisticText "COREL DRAW"
.ExtractText "C:\TEXTFILE.TXT"
```

The above example extracts the text "COREL DRAW" to a text file named "TEXTFILE.TXT".

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

FitTextToPath (DRAW)

.FitTextToPath .TextOrientation = *long*, .VertAlign = *long*, .HorizAlign = *long*, .CurveSideToFit = *long*, .FitOtherSide = *boolean*, .HorizOffset = *long*, .DistFromPath = *long*

This command fits selected artistic text to the selected path.

Parameter	Description
.TextOrientation	0 = Rotates individual characters to follow the contours of the path. 1 = The characters are not changed. 2 = Vertically skews each character, creating the impression that the text is standing upright on the path. 3 = Horizontally skews each character, creating the impression that the text is turning in toward the screen.
.VertAlign	If you specify a distamce from the path, then this parameter has no effect. 0 = Variable. Allows you to move the text off the path by dragging with the mouse. 1 = Bottom. Aligns the descender line of the text with the path. 2 = Top. Aligns the ascender line of the text with the path. 3 = Center. Centers the text vertically on the path. 4 = Baseline. Aligns the baseline of the text with the path.
.HorizAlign	1 = Aligns the text with the start node of the line or curve. 2 = Aligns the text with the end point of the line or curve. 3 = Centers the text on the path.
.CurveSideToFit	1 = Aligns the text to the top of a closed object. 2 = Aligns the text to the left of a closed object. 3 = Aligns the text to the bottom of a closed object. 4 = Aligns the text to the right of a closed object.
.FitOtherSide	Set to TRUE (-1) to place the text on the other side of the path.
.HorizOffset	Lets you specify the distance the text is offset from the start node.
.DistFromPath	Lets you specify the distance the text is from path.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

GetTextFontName (DRAW)

This command returns the font name of the selected text.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

GetTextFontSize (DRAW)

This command returns the font size of the selected text.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

GetTextString (DRAW)

This command returns the actual text string.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

GetTextWordCount (DRAW)

This command returns the number of words in your document.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

MergeBackText (DRAW)

.MergeBackText .SourceFile = *string*

This command merges the extracted text back into the CorelDRAW document.

Parameter	Description
.SourceFile	Lets you specify the name of the source file to merge.

Example

```
.CreateArtisticText "COREL DRAW"
.ExtractText "C:\TEXTFILE.TXT"
.MergeBackText "C:\TEXTFILE.TXT"
```

The above example merges the extracted text from the file "TEXTFILE.TXT" back into the DRAW document.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetArtisticText (DRAW)

.SetArtisticText .NewText = *string*

This command allows you to change selected Artistic Text text strings.

Parameter	Description
.NewText	Lets you specify the name of the new text to set.

Example

```
.FileNew
.CreateArtisticText "1"
.FilePrint
FOR i%=2 TO 10 STEP 1
.SetArtisticText i%
.FilePrint
NEXT i%
```

The above example creates the string "1" as Artistic text and then prints the document. Within the FOR...NEXT loop, the Artistic text is changed from the numbers 2 to 10. After each change in the Artistic text, the document is printed.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetBullet (DRAW)

.SetBullet .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .SymbolLibrary = *string*, .SymbolNumber = *long*, .PointSize = *long*, .BulletIndent = *long*, .VerticalShift = *long*

This command sets bullets for text.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.SymbolLibrary	Lets you specify the name of the symbol library. Refer to the Effects tab of the Paragraph dialog box for more details.
.SymbolNumber	Lets you specify the selected symbol number. Refer to the Effects tab of the Paragraph dialog box for more details.
.PointSize	Lets you specify the point size in tenths of a point.
.BulletIndent	Lets you specify the size of the bullet indentation in tenths of a micron.
.VerticalShift	Lets you specify the amount of baseline shift in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetBullet 32, 123, "Animals 1", 55, 480, 400000, 0
```

The above inserts a camel bullet, indented 1.57 inches.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetCharacterAttributes (DRAW)

.SetCharacterAttributes .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .FontName = *string*, .FontStyle = *long*, .PointSize = *long*, .Underline = *long*, .Overline = *long*, .StrikeOut = *long*, .Placement = *long*, .CharacterSpacing = *long*, .WordSpacing = *long*, .LineSpacing = *long*, .Alignment = *long*

This command sets the text character attributes.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.FontName	Lets you specify the font name.
.FontStyle	Lets you specify the style of the selected font. 7 = Normal 8 = Normal/Italic 13 = Bold 14 = Bold/Italic
.PointSize	Lets you specify the size of the selected font in tenths of a point.
.Underline	Lets you specify the type of underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
.Overline	Lets you specify the type of overline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
.StrikeOut	Lets you specify the type of strikeout. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick words 5 = Double thin 6 = Double thin words
.Placement	Lets you specify the placement of the font. 0 = Normal 1 = Superscript 2 = Subscript
.CharacterSpacing	Lets you specify the character spacing in tenths of a percent.
.WordSpacing	Lets you specify the word spacing in tenths of a percent.
.LineSpacing	Lets you specify the line spacing in tenths of a percent.
.Alignment	Lets you specify the alignment. 0 = None 1 = Left 2 = Center 3 = Right 4 = Full justify 5 = Force justify

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .SetCharacterAttributes command.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
```

```
.SelectObjectsInRect 250000, -300000, -250000, 1100000, 0
.SetCharacterAttributes 0, 4, "Arial", 13, 900, 0, 0, 0, 0, 0, 0, 0, 1
```

The above example creates the text "COREL", then sets the font to Arial, the font type to Bold, and the point size to 90.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetFrameColumn (DRAW)

.SetFrameColumn .ColumnNumber = *long*, .Width = *long*, .GutterWidth = *long*

This command formats columns for text.

Parameter	Description
.ColumnNumber	Lets you specify the column number.
.Width	Lets you specify the width of the column in tenths of a micron.
.GutterWidth	Lets you specify the width of the gutter in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command. This command must be called twice.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetFrameColumn 0, 500000, 50000
.SetFrameColumn 1, 500000, 50000
```

The above example formats the text into two columns, each 2 inches wide.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetIndents (DRAW)

.SetIndents .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .FirstLine = *long*, .RestOfLines = *long*, .RightMargin = *long*

This command sets indents for text.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.FirstLine	Lets you specify the size of the first line indentation, in tenths of a micron.
.RestOfLines	Lets you specify the size of the remaining line indentation, in tenths of a micron.
.RightMargin	Lets you specify the size of the right margin indentation, in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.
- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetIndents 0, 0, 0, 400000, 0
```

The above example indents all lines except the first by 1.57 inches.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

SetParagraphSpacing (DRAW)

.SetParagraphSpacing .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .CharacterSpacing = *long*, .WordSpacing = *long*, .LineSpacing = *long*, .BeforeParagraph = *long*, .AfterParagraph = *long*, .Alignment = *long*, .AutoHyphenation = *boolean*, .HyphenHotZone = *long*

This command sets paragraph spacing.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.CharacterSpacing	Lets you specify the character spacing in tenths of a percent.
.WordSpacing	Lets you specify the word spacing in tenths of a percent.
.LineSpacing	Lets you specify the line spacing in tenths of a percent.
.BeforeParagraph	Lets you specify the spacing before paragraphs in tenths of a percent.
.AfterParagraph	Lets you specify the spacing after paragraphs in tenths of a percent.
.Alignment	Lets you specify the alignment. 0 = None 1 = Left 2 = Center 3 = Right 4 = Full justify 5 = Force justify
.AutoHyphenation	Set to TRUE (-1) to enable automatic hyphenation. Set to FALSE (0) to disable this option.
.HyphenHotZone	Lets you specify the size of the hyphen hot zone in tenths of a micron.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 1000000, -1000000, -1000000, 1000000, "Lets you specify the type of
underline. 0 = None 1 = Single thin 2 = Single thin words 3 = Single thick 4 = Single thick
words 5 = Double thin 6 = Double thin words"
.SelectObjectsInRect 1000000, -1000000, -1000000, 1000000, 0
.SetParagraphSpacing 0, 0, 900, 900, 900, 200, 200, 1, 0, 0
```

The above example creates a text string, selects the entire text and applies paragraph spacing to it.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} **Related Topics**

SetTextString (DRAW)

.SetTextString .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .Text = *string*

This command changes the text in a selected text object.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.
.Text	Lets you specify the text. Maximum string length is 255 characters.

Note

- The .CreateTextString and .SelectObjectsInRect functions must be called before this command.

Example

```
.CreateTextString 250000, -300000, -250000, 1100000, "COREL"
.SelectObjectsInRect 250000, -300000, -250000, 1100000, 0
.SetCharacterAttributes 0, 4, "Arial", 13, 900, 0, 0, 0, 0, 0, 0, 0, 1
.SetTextString -1, -1, "RT"
.SetCharacterAttributes 5, 6, "Arial", 8, 900, 0, 0, 0, 1, 0, 0, 0, 0
```

The above example creates the text string "COREL", then appends a second text string "RT" to it. The appended string is italic and superscript.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

StraightenText (DRAW)

.StraightenText .FirstSelectedChar = *long*, .LastSelectedChar = *long*

This command resets the kerning angle to 0 for selected text.

Parameter	Description
.FirstSelectedChar	Lets you specify the starting character of the selected text. Note: The first character in a string is equal to 0, not 1.
.LastSelectedChar	Lets you specify the ending character of the selected text.

Example

```
.StraightenText 0, 0
```

The above example straightens the first character of selected paragraph text.

{button ,AL(`OVR1 Text commands;',0,"Defaultoverview",)} Related Topics

Fill and outline commands

AddArrowPoint (DRAW)

.AddArrowPoint .X = *long*, .Y = *long*, .Enabled = *boolean*, .Letter = *boolean*, .User = *boolean*, .Closed = *boolean*, .Continuity = *long*, .NodeType = *long*

This command adds a node that is part of an arrowhead. This command must be part of a block of commands that begins with the .BeginDrawArrow command and ends with the .EndDrawArrow command. To add the arrowhead to a path, use the .SetOutlineArrow command.

Parameter	Description
.X	Lets you specify the X-coordinate of the node in tenths of a micron, relative to the center of the arrowhead area.
.Y	Lets you specify the Y-coordinate of the node in tenths of a micron, relative to the center of the arrowhead area.
.Enabled	Always set to FALSE (0)
.Letter	Always set to FALSE (0)
.User	Set to TRUE (-1) to make this node selectable.
.Closed	Set to TRUE (-1) to make the arrowhead closed.
.Continuity	0 = cusp node 1 = smooth node 2 = symmetrical node
.NodeType	1 = straight line segment 2 = curved line segment

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyFountainFill (DRAW)

.ApplyFountainFill .Type = *long*, .CenterX = *long*, .CenterY = *long*, .Angle = *long*, .Steps = *long*, .Padding = *long*, .Blend = *long*, .Rate = *long*

This command lets you apply a Fountain Fill to the selected object. If the Blend was Custom, then all intermediate colors will be lost unless the Blend applied is again Custom. If the existing fill is not fountain, the start color will be CMYK Black and the end color CMYK white.

Parameter	Description
.Type	Lets you specify the type of Fountain Fill to apply: 0 = Linear (default) 1 = Radial 2 = Conical 3 = Square
.CenterX	Lets you specify the horizontal offset of the center of the fill. Valid values range from -100 to 100 percent. A value of -50% will place the center on the left edge of your object; a value of 50% will place it on the right edge.
.CenterY	Lets you specify the vertical offset of the center of the fill. Valid values range from -100 to 100 percent. A value of -50% will place the center on the bottom edge of your object; a value of 50% will place it on the top edge.
.Angle	Lets you specify the angle at which the fill is applied in tenths of degrees. Positive values will rotate the fill counter-clockwise, negative values will rotate it clockwise.
.Steps	Lets you specify the number of steps you want. Lower values produce coarser fountains on screen which take less time to redraw.
.Padding	Lets you specify the amount of padding to apply to the fill. Ignored for type 2. Valid values range from 0 to 45 percent.
.Blend	Lets you specify the type of blending to apply to the fill. 0 = Direct (default) 1 = Rainbow CW 2 = Rainbow CCW 3 = Custom
.Rate	Lets you specify the mid-point used to apply the fill. Valid values range from 1 to 99.

Note

- The Horizontal and Vertical Offset options are not available for linear fountain fills; set parameters to 0.
- The Angle option is not available for circular fountain fills; set parameter to 0.
- You can use the ANGLECONVERT function to specify angle measurements
- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyFountainFill command.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.ApplyFountainFill 1, -50, -50, 900, 20, 20, 2, 1
```

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyFullColorFill (DRAW)

.ApplyFullColorFill .FileName = *string*, .TileWidth = *long*, .TileHeight = *long*, .FirstTileOffsetX = *long*, .FirstTileOffsetY = *long*, .RowOffset = *boolean*, .RowColumnOffset = *long*, .SeamlessTiling = *boolean*, .ScaleWithObject = *boolean*[, .VectorBBoxTop = *long*][, .VectorBBoxBottom = *long*][, .VectorBBoxLeft = *long*][, .VectorBBoxRight = *long*][, .RotationAngle = *long*][, SkewAngle = *long*]

This command lets you apply a Full-Color fill to the selected object.

Parameter	Description
.FileName	Lets you specify the name of the Fill file.
.TileWidth	If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
.TileHeight	If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
.FirstTileOffsetX	X offset from center of object in the same units used by width & height.
.FirstTileOffsetY	Y offset from center of object in the same units used by widht & height.
.RowOffset	Set to TRUE (-1) to enable row offset. Set to FALSE (0) to enable column offset.
.RowColumnOffset	Lets you specify the amount of row or column offsets. Valid values range from 0 to 100.
.SeamlessTiling	Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
.ScaleWithObject	Set to TRUE (-1) to scale the pattern with the object. Set to FALSE (0) to disable this option.
.VectorBBoxTop (optional)	Lets you specify the top coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
.VectorBBoxBottom (optional)	Lets you specify the bottom coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
.VectroBBoxLeft (optional)	Lets you specify the left coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
.VectorBBoxRight (optional)	Lets you specify the right coordinate of the fill's bounding box in tenths of a micron, relative to the center of the page.
.Rotation Angle (optional)	Lets you specify the amount in which the tile is rotated in millionths of degrees.
.SkewAngle (optional)	Lets you specify the amount in which the tile is skewed in millionths of degrees.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyFullColorFill "C:\MONTEMP.BMP", 500000, 500000, 100, 100, 0, 100, 0, 0
```

The above example applies a full color fill to a rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyNoFill (DRAW)

.ApplyNoFill

This command removes the fill from the selected object, allowing objects behind it to show through.

Example

```
.SelectAllObjects
.ApplyNoFill
```

The above example removes the fill from all objects.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyOutline (DRAW)

.ApplyOutline .Width = *long*, .Type = *long*, .EndCaps = *long*, .JoinType = *long*, .Aspect = *long*, .Angle = *long*, .DotDash = *long*, .RightArrow = *long*, .LeftArrow = *long*, .BehindFill = *boolean*, .OutlineType = *long*, .Preset = *long*, .ScalePen = *boolean*

This command lets you apply an Outline to the selected object.

Parameter	Description
.Width	Lets you specify the width of the outline to apply, in tenths of a micron.
.Type	Lets you specify the outline type: 0 = None 1 = Solid 2 = Dot - Dash
.EndCaps	Lets you specify the end caps to be applied to the outline: 0 = Butt 1 = Round 2 = Square
.JoinType	Lets you specify the outline join types: 0 = Miter 1 = Round 2 = Bevel
.Aspect	Lets you specify the stretch field which adjusts the width of the nib. Valid values range from 1 to 100 percent.
.Angle	Lets you specify the angle of the nib's edge, in tenths of degrees.
.DotDash	Lets you specify the type of dot/dash line. Dot/dash line types are listed in the Style drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list the first listed type is identified as 0, the second listed type is identified as 1, and so on.
.RightArrow	Lets you specify the style of right-arrow. Right-arrow types are listed in the right arrow drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list (from left-to-right) the first listed type is identified as 0, the second listed type is identified as 1, and so on.
.LeftArrow	Lets you specify the style of left-arrow. Left-arrow types are listed in the left arrow drop-down list box of the Outline Pen Roll-Up. The types are numbered and identified according to their position in the list (from left-to-right) the first listed type is identified as 0, the second listed type is identified as 1, and so on.
.BehindFill	Set to TRUE (-1) to position the outline behind the fill. Set to FALSE (0) to position the outline in front of the fill.
.OutlineType	Lets you specify the type of preset outline. 0 = Pen 1 = Outline 2 = PenOutline
.Preset	Lets you specify the tint of the outline. Values range from 1 (0%) to 11 (100%). A value of 0 has no effect on the outline. This parameter is only used if the selected outline type supports preset tints.
.ScalePen	Set to TRUE (-1) to scale the outline when the object is scaled.

Note

- You can use the ANGLECONVERT function to specify angle measurements
- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyOutline command.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyOutline 50000, 2, 0, 1, 50, 250, 2, 0, 0, 0
```

The above example applies a dashed outline 50000 microns wide, with round corners to the rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyPostscriptFill (DRAW)

.ApplyPostscriptFill .PSFill = *string*, .NumParms = *long*, .Parm1 = *long*, .Parm2 = *long*, .Parm3 = *long*, .Parm4 = *long*, .Parm5 = *long*

This command lets you apply a PostScript Fill to a selected object.

Parameter	Description
.PSFill	Lets you specify the name of the postscript fill. The name must be preceded by an F/. For a listing of Postscript fills available, see the PostScript Texture dialog box. If you create custom PostScript fills, their definitions are placed in the USERPROC.PS file in the Custom folder of your Corel folder. The PostScript fills definitions supplied with DRAW are also in this file.
.NumParms	Lets you specify the number of parameters used for the selected PostScript Fill, an integer value between 1 and 5, inclusive. Refer to the PostScript Texture dialog box to determine the number of parameters for a full fill. Set to 2 for spot fills.
.Parm1	Lets you specify the first parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If you're using a spot fill, set Parm1 to a value between -1 and 1.
.Parm2	Lets you specify the second parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set Parm2 to a value between -1 and 1.
.Parm3	Lets you specify the third parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.
.Parm4	Lets you specify the fourth parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.
.Parm5	Lets you specify the fifth parameter for the selected PostScript Fill. This parameter varies depending on the Fill selected. Refer to the PostScript Texture dialog box for full fill parameter specifics. If the parameter is not used with the fill you selected, set it to 0. If you're using a spot fill, set this parameter to 0.

Note

- If you create custom PostScript fills, their definitions are placed in the USERPROC.PS file in the Custom folder of your Corel folder. The PostScript fills definitions supplied with CorelDRAW are also in this file.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyPostScriptFill "F/StoneWall", 4, 15, 100,0, 5, 0
```

The above example applies the StoneWall PostScript fill to the selected rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyPreset (DRAW)

.ApplyPreset .PresetFileName = *string*, .PresetName = *string*

This command lets you load and apply a Preset.

Parameter	Description
.PresetFileName	Lets you specify the name of the Preset File.
.PresetName	Lets you specify the name of the Preset.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyPreset "C:\CORELDRW.PST", "Button Blue"
```

The above example applies the specified preset fill to the rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyTextureFill (DRAW)

.ApplyTextureFill .TextureLibrary = *string*, .TextureName = *string*, .TextureStyle = *string,* .TextureWidth = *long*, .TextureHeight = *long*, .TextureOffsetX = *long*, .TextureOffsetY = *long*, .RowOffset = *boolean*, .RowColumnOffset = *long*, .ScaleWithObject = *boolean*, .RotationAngle = *long*, .SkewAngle = *long*

This command lets you apply one of the texture fills included in CorelDRAW.

Parameter	Description
.TextureLibrary	Lets you specify the name of the Texture Library.
.TextureName	Lets you specify the name of the texture.
.TextureStyle	Lets you specify the name of the style. If you set .TextureLibrary to "Samples 5", the style name must be preceded by "CDR5:". For example, "CDR5:Blue Valley".
.TextureWidth	If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
.TextureHeight	If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
.TextureOffsetX	X offset from center of object in the same units used by width & height.
.TextureOffsetY	Y offset from center of object in the same units used by widht & height.
.RowOffset	Set to TRUE enables the row offset, Set to FALSE enables the column offset.
.RowColumnOffset	Lets you specify the amount that the row or columns is offset as a percentage.
.ScaleWithObject	Set to TRUE will transform the fill with the object.
.RotationAngle	Lets you specify the amount that the texture is rotated in millionths of degrees.
.SkewAngle	Lets you specify the amount that the texture is skewed in millionths of degrees.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyTextureFill "Styles", "Satellite Photography", "Satellite Photography"
```

The above example creates a rectangle, then applies the satellite photography fill to it.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ApplyTwoColorFill (DRAW)

.ApplyTwoColorFill .FileName = *string*, .TileWidth = *long*, .TileHeight = *long*, .FirstTileOffsetX = *long*, .FirstTileOffsetY = *long*, .RowOffset = *boolean*, .RowColumnOffset = *long*, .SeamlessTiling = *boolean*, .ScaleWithObject = *boolean,* .RotationAngle = *long,* .SkewAngle = *long*

This command lets you apply a Two-Color fill to the selected object.

Parameter	Description
.FileName	Lets you specify the name of the two color fill file to use. See the Two-Color Bitmap Pattern dialog box for a list of valid file formats.
.TileWidth	If less than 500 and ScaleWithObject is set, is a percentage of the object width, otherwise is in tenths of a micron.
.TileHeight	If less than 500 and ScaleWithObject is set, is a percentage of the object height, otherwise is in tenths of a micron.
.FirstTileOffsetX	X offset from center of object in the same units used by width & height.
.FirstTileOffsetY	Y offset from center of object in the same units used by widht & height.
.RowOffset	Set to TRUE (-1) to enable row offset. Set to FALSE (0) to enable column offset.
.RowColumnOffset	Lets you specify the amount of row or column offsets. Valid values range from 0 to 100.
.SeamlessTiling	Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
.ScaleWithObject	Set to TRUE (-1) to enable seamless tiling. Set to FALSE (0) to disable this option.
.Rotation Angle	Lets you specify the amount in which the tile is rotated in millionths of degrees.
.SkewAngle	Lets you specify the amount in which the tile is skewed in millionths of degrees.

Note

- You can use the LENGTHCONVERT function, or one of the FROM... or TO... functions to specify length measurements.

Example

```
.CreateRectangle 1000000, -500000, -1000000, 500000, 0
.ApplyTwoColorFill "mybitmap.bmp", 5, 255, 0, 0, 0, 5, 0, 0, 0, 0, 500000, 500000, 100, 100,
0, 100, 0, 0, 0
```

The above example applies a two-color bitmap fill from the MYBITMAP.BMP file to the rectangle.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

ApplyUniformFillColor (DRAW)

.ApplyUniformFillColor

This command lets you apply a Uniform Fill Color to a selected object.

Example

```
.CreateEllipse -250000, -500000, 250000, 500000, 0, 0, 0
.StoreColor DRAW_COLORMODEL_CMYK, 100, 100, 0, 0, 0
.ApplyUniformFillColor
```

The above example creates an ellipse and uniformly fills it with cyan.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

BeginDrawArrow (DRAW)

.BeginDrawArrow .LeftArrow = *boolean*, .LineOffset = *long*, .NumOfPoints = *long*

This command initializes a block of arrowhead creation commands. This block must include one or more instances of the .AddArrowPoint command and this block must end with the .EndDrawArrow command. To add the arrowhead to a path, use the .SetOutlineArrow command.

Parameter	Description
.LeftArrow	Set to TRUE (-1) to create a left-facing arrowhead. Set to FALSE (0) to create a right-facing arrowhead.
.LineOffset	Lets you specify the distance between the end of the path and the arrowhead.
.NumOfPoints	Lets you specify the number of nodes in your arrowhead.

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

ConvertColor (DRAW)

.ConvertColor .ToColorModel = *long*, .V1 = *long*, .V2 = *long*, .V3 = *long*, .V4 = *long*, .V5 = *long*, .V6 = *long*

This command converts one color model to another color model.

Parameter	Description
.ToColorModel	Lets you specify the destination folder.
.V1	Returns the first color component.
.V2	Returns the second color component.
.V3	Returns the third color component.
.V4	Returns the fourth color component.
.V5	Returns the fifth color component.
.V6	Returns the sixth color component.
.V7	Returns the color density from 0 to 100

Example

```
.GetUniformFillColor InModel, In1, In2, In3, In4, In5, In6, In7
.StoreColor InModel, In1, In2, In3, In4, In5, In6, In7
.ConvertColor DRAW_COLORMODEL_RGB, RedVal, GREENVAL, BLUEVAL, 0, 0, 0, 0
```

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

CreatePaletteFromDocument (DRAW)

.CreatePaletteFromDocument .Filename = *string*, .Overwrite = *boolean*

This command creates a color palette from the current document.

Parameter	Description
.Filename	Lets you specify the file name for the color palette.
.Overwrite	Set to TRUE (-1) overwrites the specifief file name.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

CreatePaletteFromSelection (DRAW)

.CreatePaletteFromSelection .Filename = *string*, .Overwrite = *boolean*

This command creates a color palette from the current selection.

Parameter	Description
.Filename	Lets you specify the file name for the color palette.
.Overwrite	Set to TRUE (-1) overwrites the specifief file name.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

DeletePaletteColor (DRAW)

.DeletePaletteColor .Index = *long*

This command deletes the palette color from the default palette.

Parameter	Description
.Index	Lets you specify the color index that you want to delete.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

EndDrawArrow (DRAW)

.EndDrawArrow

This command ends a block of arrowhead creation commands. This block must include one or more instances of the .AddArrowPoint command and this block must begin with the .BeginDrawArrow command.

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

GetColor (DRAW)

.GetColor .StoreColor = *boolean*[, .ColorModel = *long*][, .V1 = *long*][, .V2 = *long*][, .V3 = *long*][, .V4 = *long*][, .V5 = *long*][, .V6 = *long*][, .V7 = *long*]

This function gets a color from the color dialog box.

Parameter	Description
.StoreColor	-1 stores the color 0 does not store the color
.ColorModel (optional)	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1 (optional)	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click + for valid value ranges.
.Color2 (optional)	Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.Color3 (optional)	Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.Color4 (optional)	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V5 (optional)	Lets you specify the fifth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V6 (optional)	Lets you specify the sixth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V7 (optional)	Lets you specify the seventh color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetCurrentPaletteName (DRAW)

.GetCurrentPaletteName

This function returns the name of the current color palette.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

GetFillType (DRAW)

ReturnValue& = .GetFillType()

This function returns the Fill Type of a selected object. If more than one object is selected, the function returns the Fill Type of the last selected object.

Return Value

Returns one of the following values:

- 0 None
- 1 Uniform
- 2 Fountain
- 6 PostScript
- 7 Two-color
- 9 ColorBitmap
- 10 Vector
- 11 Texture

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .GetFillType command.

Example

```
.SelectObjectOfCDRStaticID IDRect&
fillType& = .GetFillType()
Message fillType&
```

The above example displays a number corresponding to the fill type of the selected object in a message box.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetFountainFill (DRAW)

.GetFountainFill .Type = *long**, .CenterX = *long**, .CenterY = *long**, .Angle = *long**, .Steps = *long**, .Padding = *long**, .Blend = *long**, .Rate = *long**, .NumColors = *long**

This command returns the Fountain Fill attributes of a selected object. If more than one object is selected, the function returns the Fountain Fill attributes of the last selected object.

Parameter	Description
.Type	Returns the type of Fountain Fill: 0 = Linear (default) 1 = Radial 2 = Conical 3 = Square
.CenterX	Returns the Horizontal Offset of the center of the fill. Valid values range from -100 to +100 percent. A value of -50% will place the center on the left edge of your object; a value of 50% will place it on the right edge.
.CenterY	Returns the Horizontal Offset of the center of the fill. Valid values range from -100 to +100 percent. A value of -50% will place the center on the bottom edge of your object; a value of 50% will place it on the top edge.
.Angle	Returns the angle at which the fill is applied in degrees. Positive values will rotate the fill counter-clockwise, negative values will rotate it clockwise.
.Steps	Returns the number of stripes you want. Lower values produce coarser fountains on screen which take less time to redraw. Valid values range from 2 to 256.
.Padding	Returns the amount of padding to apply to the fill. Ignored for type 2. Valid values range from 0 to 45 percent.
.Blend	Returns the type of blending to apply to the fill. 0 = Direct (default) 1 = Rainbow CW 2 = Rainbow CCW 3 = Custom
.Rate	Returns the rate method used to apply the fill.
.NumColors	Returns the number of colors.

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.GetFountainFill fillType&, CX&, CY&, Angle&, Steps&, Pad&, Blend&, Rate&, Num&
MESSAGE fillType&
```

The above example returns Fountain Fill attributes and displays a number corresponding to the fill type in a message box.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetFountainFillColor (DRAW)

.GetFountainFillColor .Index = *long*, .Position = *long**, .ColorModel = *long**, .V1 = *long**, .V2 = *long**, .V3 = *long**, .V4 = *long*,* .V5 = *long*,* .V6 = *long*,* Density = *long**

This command Retrieves a color from a fountain fill.

Parameter	Description
.Index	The index number of the color you want to get. Use the .GetFountainFill command to find out how many colors are in a fountain fill. Valid index numbers will range from 0 to the number of colors minus 1. If you use the value 100, you will always get the end color.
.Position	Return the position of the color within the fill.
.ColorModel	Returns the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome
.V1	Returns the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.
.V2	Returns the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V3	Returns the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V4	Returns the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.
.V5	Returns the fifth color component for .plColorModel.
.V6	Returns the sixth color component for .plColorModel.
.Density	Returns the color density.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

GetOutline (DRAW)

.GetOutline .Width = *long**, .Type = *long**, .EndCaps = *long**, .JoinType = *long**, .Aspect = *long**, .Angle = *long**, .DotDash = *long**, .RightArrow = *long**, .LeftArrow = *long**, .BehindFill = *boolean**, .ScalePen = *boolean**

This function returns the outline attributes of the selected object. If more than one object is selected, the function returns the outline attributes of the last selected object.

Parameter	Description
.Width	Returns the width of the outline, in tenths of a micron.
.Type	Returns the outline type: 0 = None 1 = Solid 2 = Dot - Dash
.EndCaps	Returns the End Caps applied to the outline: 0 = Butt 1 = Round 2 = Square
.JoinType	Returns the outline join types: 0 = Miter 1 = Round 2 = Bevel
.Aspect	Returns the stretch field which adjusts the width of the nib.
.Angle	Returns the angle of the nib's edge, in tenths of degrees.
.DotDash	Returns the type of dot/dash line. Refer to the Outline Pen dialog box for more details.
.RightArrow	Returns the style of right-arrow. Refer to the Outline Pen dialog box for more details.
.LeftArrow	Returns the style of left-arrow. Refer to the Outline Pen dialog box for more details.
.BehindFill	Returns the position of the outline fill. TRUE (-1) = Outline behind fill FALSE (0) = Outline in front of fill
.ScalePen	Returns the the scale pen setting. TRUE (-1) = Outline is scaled when object is scaled FALSE (0) = Outline is not scaled when object is scaled

Note

- You can use the ANGLECONVERT function to specify angle measurements

Example

```
.GetOutline Width&, outlineType&, EndCaps&, JoinType&, Aspect&, Angle&, DotDash&, RArrow&,
LArrow&, BehindFill&
```

The above example returns the outline attributes of the selected object.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetOutlineColor (DRAW)

.GetOutlineColor .ColorModel = *long**, .Color1 = *long**, .Color2 = *long**, .Color3 = *long**, .Color4 = *long**, .Color5 = *long**, .Color6 = *long**, Density = *long*

This function returns the Outline Color attributes of a selected object. If more than one object is selected, the function returns the Outline Color attributes of the last selected object.

Parameter	Description
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Returns the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.

.Color2 Returns the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges.

.Color3 Returns the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges.

.Color4	Returns the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges.

.Color5 Lets you specify the fifth color component for .ColorModel.

.Color6 Lets you specify the sixth color component for .ColorModel.

.Density Lets you specify the color density from 0-100.

Example

```
.GetOutlineColor Model&, C1&, C2&, C3&, C4&, C5&, C6&, C7&
MESSAGE Model&
```

The above example determines the outline color attributes of the selected object and displays a number corresponding to the color model in a message box.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetPaletteColor (DRAW)

.GetPaletteColor .Index = *long*, .ColorModel = *long*, .Color1 = *long*, .Color2 = *long*, .Color3 = *long*, .Color4 = *long*, .Color5 = *long*, .Color6 = *long,* .Color7 = *long*

This function returns the palette color attributes of a selected object.

Parameter	Description
.Index	Lets you specify the color index.
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.

.Color2 Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color3 Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color4	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color5 Lets you specify the fifth color component for .ColorModel.

.Color6 Lets you specify the sixth color component for .ColorModel.

.Color7 Lets you specify the seventh color component (density)

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

GetPaletteColorCount (DRAW)

.GetPaletteColorCount

This function returns the number of colors in the default Color Palette.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

GetPaletteColorName (DRAW)

.GetPaletteColorName .Index = *long*

This command retrieves the Color name for a specific index.

Parameter	Description
.Index	Lets you specify the index of the color for which you want to retrieve the name.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

GetUniformFillColor (DRAW)

.GetUniformFillColor .ColorModel = *long*, .Color1 = *long*, .Color2 = *long*, .Color3 = *long*, .Color4 = *long*, .Color5 = *long*, .Color6 = *long,* .Color7 = *long*

This function returns the Uniform Fill color attributes of a selected object. If more than one object is selected, the function returns the Uniform Fill color of the last selected object.

Parameter	Description
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.

.Color2 Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color3 Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color4	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color5 Lets you specify the fifth color component for .ColorModel.

.Color6 Lets you specify the sixth color component for .ColorModel.

.Color7 Lets you specify the seventh color component (density)

Example

```
.GetUniformFillColor Model&, C1&, C2&, C3&, C4&, C5&, C6&, C7&
MESSAGE Model&
```

The above example determines the uniform fill color of the selected object and displays a number corresponding to the color model in a message box.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

InsertPaletteColor (DRAW)

.InsertPaletteColor .Index = *long*, .Name = *string*, .ColorModel = *long*, .Color1 = *long*, .Color2 = *long*, .Color3 = *long*, .Color4 = *long*, .Color5 = *long*, .Color6 = *long,* .Color7 = *long*

This command inserts a color into the default palette.

Parameter	Description
.Index	Lets you specify the color index.
.Name	Lets you specify the color name.
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.

.Color2 Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color3 Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color4	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color5 Lets you specify the fifth color component for .ColorModel.

.Color6 Lets you specify the sixth color component for .ColorModel.

.Color7 Lets you specify the seventh color component (density)

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

LoadPalette (DRAW)

.LoadPalette .Filename = *string*

This command loads the specifies color palette in to the current document.

Parameter	Description
.Filename	Lets you specify the file name of the color palette you want to load in to your document.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

OverPrintFill (DRAW)

.OverPrintFill

This command overprints the fill of the selected object.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

OverPrintOutline (DRAW)

.OverPrintOutline

This command overprints the outline of the selected object.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

RemoveFountainFillColor (DRAW)

.RemoveFountainFillColor .Position = *long*

This command removes the currently selected Fountain Fill Color.

Parameter	Description
.Position	Lets you specify the position of the color to be removed. 0 and 100 are invalid values. For any other value, the color at that position is removed, if one exists. Existing fill must be a Fountain and Blend must be custom.

Example

```
.ApplyFountainFill 2, -50, -50, 900, 20, 20, 2, 0
.RemoveFountainFillColor 75
```

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

ReplacePaletteColor (DRAW)

.ReplacePaletteColor .Index = *long*, .Name = *string*, .ColorModel = *long*, .Color1 = *long*, .Color2 = *long*, .Color3 = *long*, .Color4 = *long*, .Color5 = *long*, .Color6 = *long,* .Color7 = *long*

This command replaces the color in the default palette.

Parameter	Description
.Index	Lets you specify the color index.
.Name	Lets you specify the color name.
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▶ for valid value ranges.

.Color2 Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color3 Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color4	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▶ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color5 Lets you specify the fifth color component for .ColorModel.

.Color6 Lets you specify the sixth color component for .ColorModel.

.Color7 Lets you specify the seventh color component (density)

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

SavePalette (DRAW)

.SavePalette .Filename = *string*, .Overwrite = *boolean*

This command saves the current color palette.

Parameter	Description
.Filename	Lets you specify the file name for the color palette.
.Overwrite	Set to TRUE (-1) overwrites the existing palette.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

SetOutlineArrow (DRAW)

.SetOutlineArrow .ArrowType = *long*

This command changes the arrowhead of the outline of the selected object. This command follows a block of arrowhead creation commands (see example).

Parameter	Description
.ArrowType	Lets you specify the arrow position. 0 = left arrow 1 = right arrow 2 = both arrows

Example

```
.BeginDrawArrow TRUE, 0, 7
.AddArrowPoint -1016000, 0, FALSE, FALSE, TRUE, TRUE, 0, 0
.AddArrowPoint -481838, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, -271272, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 31750, 0, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint 571500, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -481838, 299974, FALSE, FALSE, TRUE, FALSE, 0, 1
.AddArrowPoint -1016000, 0, FALSE, FALSE, FALSE, TRUE, 0, 1
.EndDrawArrow
.SetOutlineArrow 0
```

The above example adds an arrowhead to the selected line.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

SetOutlineColor (DRAW)

.SetOutlineColor

This command sets the color to be applied to the outline.

Example

```
.StoreColor DRAW_COLOR_CMYK100, 0, 0, 255, 0
.SetOutlineColor
```

The above example sets the outline color to yellow.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

SetOutlineMiscProperties (DRAW)

.SetOutlineMiscProperties .Type = *long,*.Style = *long,* .Corners = *long,* .LineCaps = *long,* .Aspect = *long,* .Angle = *long,* .BehindFill = *Boolean,* .ScalePen = *boolean*

This command sets the outline properties of the selected object.

Parameter	Description
.Type	Lets you specify the outline type: 0 = None 1 = Solid 2 = Dot - Dash
.Style	Calls .DotDash from the ApplyOutline command.
.Corners	0 = Mitered
	1 = Beveled
	2 = Rounded
.LineCaps	0 = Square
	1 = Rounded
	2 = Extended Square
.Aspect	Lets you specify the stretch field which adjusts the width of the nib. Valid values range from 1 to 100 percent.
.Angle	Lets you specify the angle of the nib's edge, in tenths of degrees.
.BehindFill	Set to TRUE (-1) to position the outline behind the fill. Set to FALSE (0) to position the outline in front of the fill.
.ScalePen	Set to TRUE (-1) to scale the outline when the object is scaled.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

SetOutlineWidth (DRAW)

.SetOutlineWidth .Width = *long*

This command changes the width of the outline of the selected object.

Parameter	Description
.Width	Lets you specify the width of the outline.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} **Related Topics**

StoreColor (DRAW)

.StoreColor .ColorModel = *long*, .Color1 = *long*, .Color2 = *long*, .Color3 = *long*, .Color4 = *long,*.Color5 = *long,* .Color6 = *long,* .Color7 = *long*

This command sets the color that will be applied to the selected object.

Parameter	Description
.ColorModel	Lets you specify the Color Model to use: 1 = Pantone 2 = CMYK100 3 = CMYK255 4 = CMY 5 = RGB 6 = HSB 7 = HLS 8 = BW 9 = Gray 11 = YIQ255 12 = LAB 13=Index 14=Pantone Hex 15=Hexachrome To use a color palette with the color model, multiply the palette value (listed below) by 1000 and add it to the color model value. Color palettes: 1 = TRUMATCH 2 = PANTONE PROCESS 3 = PANTONE SPOT 4 = IMAGE 5 = USER 6 = CUSTOMFIXED 7 = RGBSTANDARD 8 = FOCOLTONE 9 = DUPONT 10 = TOYO 11 = DIC 12 = PANTONE HEX 13 = LAB 14 = NETSCAPE 15 = EXPLORER 16 = USERINKS
.Color1	Lets you specify the first color component for .ColorModel. For example, Hue is the first color component for HSB. Click ▸ for valid value ranges.

.Color2 Lets you specify the second color component for .ColorModel. For example, Green is the second color component for RGB. Click ▸ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color3 Lets you specify the third color component for .ColorModel. For example, Saturation is the third color component for HLS. Click ▸ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

.Color4	Lets you specify the fourth color component for .ColorModel. For example, Black is the fourth color component for CMYK. Click ▸ for valid value ranges. If this parameter is not available in the Color Model specified, set it to 0.

Example

```
.StoreColor DRAW_COLORMODEL_CMYK255, 0, 255, 0, 0, 0, 0, 0
.SetOutlineColor
```

The above example stores the Magenta color for use in the SetOutlineColor command.

{button ,AL(`OVR1 Fill and outline commands;',0,"Defaultoverview",)} Related Topics

Special effects commands

AddEnvelopeEffect (DRAW)

.AddEnvelopeEffect .PresetNumber = *long*, .MappingMode = *long*, .KeepLines = *boolean*

This command adds an envelope to the selected object.

Parameter	Description
.PresetNumber	Lets you specify the envelope preset number. Choose from 1 to 39.
.MappingMode	Lets you specify the mapping mode. 0 = Horinzontal 1 = Original 2 = Putty 3 = Vertical
.KeepLines	Set to TRUE (-1) keeps the lines of the source object. Set to FALSE (0) excludes the lines of the source object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyBitmapEffect (DRAW)

.ApplyBitmapEffect .EffectID = *long*

This command applies the bitmap effect to the selected image.

Parameter	Description
.EffectID	Lets you specify effect ID.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyBlend (DRAW)

.ApplyBlend .Steps = *boolean*, .NoOfSteps = *long*, .AngleOfRotation = *long*, .Loop = *boolean*, .PathObjectID = *long*, .FullPath = *boolean*, .RotateAll = *boolean*, .ColorWheelMode = *long*, .MapNodeStartObject = *long*, .MapNodeEndObject = *long*, .LinearBlend = *boolean*, .LinearSpacing = *boolean*, .LinkAcceleration = *boolean*, .AccelShapes = *boolean*, .BendLogBase = *long*, .SpacingLogBase = *long*, .BlendID = *long*, .BlendType = *long*

This command applies a blend to two selected objects. The parameters below correspond to the controls in the Blend Roll-Up.

Parameter	Description
.Steps	Set to TRUE (-1) to set the number of steps. Set to FALSE (0) if the blend is on a path and you want to use fixed spacing along that path.
.NoOfSteps	Lets you specify the number of intermediate steps.
.AngleOfRotation	Lets you specify the rotation of the intermediate steps in millionths of a degree (e.g., 5000000 = 5 degrees).
.Loop	Set to TRUE (-1) to enable the loop option. Set to FALSE (0) to disable the loop option.
.PathObjectID	Lets you specify the object ID of the path object. Use .GetObjectsCDRStaticID to get an object's ID.
.FullPath	Set to TRUE (-1) to enable the blend along full path option. Set to FALSE (0) to disable the blend along full path option.
.RotateAll	Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
.ColorWheelMode	0 = straight 1 = clockwise 2 = counter-clockwise
.MapNodeStartObject	Lets you specify a node on the start object to map to a specific node on the end object. The value can range from 0 (the first node) to the number of nodes minus 1.
.MapNodeEndObject	Lets you specify a node on the end object to map to a specific node on the start object. The value can range from 0 (the first node) to the number of nodes minus 1.
.LinearBlend	Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
.LinearSpacing	Set to TRUE (-1) to enable the rotate all objects option. Set to FALSE (0) to disable the rotate all objects option.
.LinkAcceleration	Set to TRUE (-1) to link the blend acceleration options.
.AccelShapes	Set to TRUE (-1) to accelerate the change in size between the start and end objects.
.BendLogBase	Lets you specify the rate of color acceleration.
.SpacingLogBase	Lets you specify the rate of spacing acceleration.
.BlendID	Reserved for future use.
.BlendType	Reserved for future use.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyContour (DRAW)

.ApplyContour .ContourType = *long*, .Offset = *long*, .Steps = *long*, .ColorWheelDirection = *long*

This command applies a contour to the selected object.

Parameter	Description
.ContourType	0 = To Center 1 = Inside 2 = Outside
.Offset	Lets you specify the distance between contours in tenths of a micron.
.Steps	Lets you specify the number of steps.
.ColorWheelDirection	0 = straight 1 = clockwise 2 = counter-clockwise

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyContour command.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

ApplyDistortion (DRAW)

.ApplyDistortion .New= *boolean*, .DistortionType = *long*, .Amplitude = *long*, .Frequency = *long*, .Seed = *long*, .Angle = *long*, .CenterX = *long*, .CenterY = *long*, Flag = *long*

This command applies a contour to the selected object.

Parameter	Description
.New	Set to TRUE will add to existing distortion. Set to FALSE clears the existing distortion.
.DistortionType	1=Push & Pull
	2=Zipper
	3=Twister
.Amplitude	1=-200-200
	2=0-100
	3=N/A
.Frequency	1=N/A
	2=Peaks/Curve
	3=N/A
.Seed	If the .Random flag is set then .Seed sets the number of seeds for the zipper distortion.
.Angle	1=N/A
	2=N/A
	3=the maximum twist angle
.CenterX	Lets you specify the offset in X from the center of the distortion.
.CenterY	Lets you specify the offset in Y from the center of the distortion.
.Flag	1=the smoothness of the zipper
	2=the random amplitude of the zipper
	4=set the acceleration of the distortion towards the center of the zipper.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyDropShadow (DRAW)

.ApplyDropShadow .HorizOffset = *long*, .VertOffset = *long*, .Opacity = *long*, .Feather = *long*, .FeatherType = *long*, .FeatherEdge = *long*

This command applies a drop shadow to the selected object.

Parameter	Description
.HorizOffset	Lets you specify the horizontal offset in tenths of a micron.
.VertOffset	Lets you specify the vertical offset in tenths of a micron.
.Opacity	Lets you specify the shadow opacity from 0 to 100.
.Feather	Lets you specify the shadow feathering from 0 to 100.
.FeatherType	0=Inside
	1=Middle
	2=Outside
	3=Average
.FeatherEdge	0=Linear
	1=Square
	2=Flat
	3=Inverse square

Example

```
.StoreColor DRAW_COLORMODEL_RGB&, 0, 0, 0 'Make a black drop shadow
.ApplyDropShadow LENGTHCONVERT(LC_INCHES, LC_TENTHS_OFA_MICRON, 1), \\LENGHTCONVERT(LC_INCHES, LC_TENTHS_OFA_MICRON, 1), 50, 100, 0, 0
```

The above example applies a drop shadow.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyEnvelopeFrom (DRAW)

.ApplyEnvelopeFrom .ObjectID = *long*, .Mappingmode = *long*, .KeepLines = *boolean*

This command applies an envelope to the selected object from the shape of another object.

Parameter	Description
.ObjectID	Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.
.Mappingmode	0 = Horizontal 1 = Original 2 = Putty 3 = Vertical
.KeepLines	Set to TRUE (-1) to keep the lines of the source object. Set to FALSE (0) to exclude these lines.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyExtrude (DRAW)

.ApplyExtrude .ExtrudeType = *long*, .VPProperties = *long*, .CopyObjectID = *long*, .Depth = *long*, .VPHorizPos = *long*, .VPVertPos = *long*, .PageOrigin = *boolean*, .Light1Pos = *long*, .Light1Intensity = *long*, .Light2Pos = *long*, .Light2Intensity = *long*, .Light3Pos = *long*, .Light3Intensity = *long*, .FillType = *long,* .DrapeFill = *long*

This command extrudes the selected object.

Parameter	Description
.ExtrudeType	0 = Small Back 1 = Small Front 2 = Big Back 3 = Big Front 4 = Back Parallel 5 = Front Parallel
.VPProperties	0 = Vanishing Point locked to object 1 = Vanishing Point locked to page 2 = Copy VP from object (specified with .lCopyObjectID) 3 = Shared VP (specified with .lCopyObjectID)
.CopyObjectID	Lets you specify the object ID of the source object for shared and copied vanishing points. Use .GetObjectsCDRStaticID to get an object's ID.
.Depth	Lets you specify the depth of the extrusion.
.VPHorizPos	Lets you specify the X-coordinate of the vanishing point in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
.VPVertPos	Lets you specify the Y-coordinate of the vanishing point in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
.PageOrigin	Set to TRUE (-1) to position the vanishing point using absolute page coordinates. Set to FALSE (0) to position the vanishing point relative to the object's position.
.Light1Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light1Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.Light2Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light2Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.Light3Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light3Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.FillType	0 = Object fill 1 = Solid fill 2 = Shade
.DrapeFill	Set to TRUE will drape the fill over the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyFountainBitmapLens (DRAW)

.ApplyFountainBitmapLens .Type = *long*,.CenterX = *long*, .CenterY = *long*, .Angle = *long*, .Steps = *long*, .Padding = *long*, .Blend = *long*, .Rate = *long*, .Starting = *long*, .Operation = *long*,.Freeze = *boolean*

This command applies a fountain bitmap lens to the selected object.

Parameter	Description
.Type	Lets you specify the fill type: 0 = Linear 1 = Radial 2 = Conical 3 = Square
.CenterX	Lets you specify the horizontal offset to the center of the fill. Choose between -100 to 100.
.CenterY	Lets you specify the vertical offset to the center of the fill. Choose between -100 to 100.
.Angle	Lets you specify the angle of the fill in tenths of a degree.
.Steps	Lets you specify the number of steps in the fill.
.Padding	Lets you specify the amount of padding to apply to the fill.
.Blend	Lets you specify the blending type: 0 = Direct 1 = Rainbow clockwise 2 = Rainbow counterclocwise 3 = Custom
.Rate	Lets you specify the mid-point to apply between the fill colors. Choose from 0 to 100.
.Starting	Lets you specify the starting transparency.
.Operation	Lets you specify which operation to perform on the selected object.
.Freeze	Set to TRUE (-1) freezes the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyLensEffect (DRAW)

.ApplyLensEffect .LensType = *long*, .Frozen = *boolean*, .RemoveFace = *boolean*, .ViewPoint = *boolean*, .VpX = *long*, .VpY = *long*, .Param1 = *long*

This command adds a lens to the selected object.

Parameter	Description
.LensType	0 = No Lens Effect 1 = Brighten 2 = Color Add 3 = Color Limit 4 = Custom Color Map 5 = Fish Eye 6 = Heat Map 7 = Invert 8 = Magnify 9 = Tinted Grayscale 10 = Transparancy 11 = Wireframe
.Frozen	Set to TRUE (-1) to enable the Frozen option.
.RemoveFace	Set to TRUE (-1) to enable the Remove Face option.
.ViewPoint	Set to TRUE (-1) to enable the View Point option.
.VpX	Lets you specify the X-coordinate of the view point in tenths of a micron.
.VpY	Lets you specify the Y-coordinate of the view point in tenths of a micron.
.Param1	This value will vary depending on the selected lens. Refer to the Lens Roll-Up for more information

Note

- You can include the SCPCONST.CSI and DRWCONST.SCI files in your script. These files define contants for all referenced parameters used with the .ApplyLensEffect command.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

ApplyPerspectiveEffect (DRAW)

.ApplyPerspectiveEffect .Handle = *long*, .PosX = *long*, .PosY = *long*

This command adds a perspective effect to the selected object.

Parameter	Description
.Handle	Lets you specify the handle of the object that is repositioned to create the perspective effect. Valid values range from 1 to 5.
.PosX	Lets you specify the X-coordinate of the new position of the handle, in tenths of a micron
.PosY	Lets you specify the Y-coordinate of the new position of the handle, in tenths of a micron

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyPresetEnvelope (DRAW)

.ApplyPresetEnvelope .PresetNumber = *long*, .Mappingmode = *long*, .KeepLines = *boolean*

This command applies a preset envelope to the selected object.

Parameter	Description
.PresetNumber	Lets you specify the preset envelope to use. Refer to the Preset Roll-Up to see which presets are available. Valid values range from 1 to 39.
.Mappingmode	0 = Horizontal 1 = Original 2 = Putty 3 = Vertical
.KeepLines	Set to TRUE (-1) to keep the lines of the envelope. Set to FALSE (0) to exclude these lines.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyRotatedExtrude (DRAW)

.ApplyRotatedExtrude .ExtrudeType = *long*, .Depth = *long*, .XRotation = *long*, .YRotation = *long*, .ZRotation = *long*, .VPHorizPos = *long*, .VPVertPos = *long*, .PageOrigin = *boolean*, .Light1Pos = *long*, .Light1Intensity = *long*, .Light2Pos = *long*, .Light2Intensity = *long*, .Light3Pos = *long*, .Light3Intensity = *long*, .FillType = *long,* .DrapeFill = *long*

This command applies a rotated extrusion to the selected object.

Parameter	Description
.ExtrudeType	0 = Small Back 1 = Small Front 2 = Big Back 3 = Big Front 4 = Back Parallel 5 = Front Parallel
.Depth	Lets you specify the depth of the extrusion.
.XRotation	Lets you specify the rotation value for the X-axis. Valid values range from 0 to 100.
.YRotation	Lets you specify the rotation value for the Y-axis. Valid values range from 0 to 100.
.ZRotation	Lets you specify the rotation value for the Z-axis. Valid values range from 0 to 100.
.VPHorizPos	Lets you specify the X-coordinate of the vanishing point, in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
.VPVertPos	Lets you specify the Y-coordinate of the vanishing point, in tenths of a micron, relative to the center of the page or the object depending on .bPageOrigin.
.PageOrigin	Set to TRUE (-1) to position the vanishing point using absolute page coordinates. Set to FALSE (0) to position the vanishing point relative to the object's position.
.Light1Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light1Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.Light2Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light2Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.Light3Pos	Lets you specify the position of the light source. Valid values range from 0 to 16.
.Light3Intensity	Lets you specify the intensity of the light source. Valid values range from 0 to 100.
.FillType	0 = Object fill 1 = Solid fill 2 = Shade
.DrapeFill	Set to TRUE will drape the fill over the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyTextureBitmapLens (DRAW)

.ApplyTextureBitmapLens .TextureLibrary = *string*,.TextureName = *string*, .TextureStyle = *string*, TextureWidth = *long*, .TextureHeight = *long*, .TextureOffsetX = *long*, .TextureOffsetY = *long*, .RowOffset = *boolean*, .RowColumnOffset = *long*, .ScaleWithObject = boolean, .RotationAngle = *long*, .SkewAngle = *long,* .Starting = *long,* .Ending = *long,* .Operation = *long,* .Freeze = *long*

This command applies a texture bitmap lens to the selected object.

Parameter	Description
.TextureLibrary	Lets you specify the name of the texture library.
.TextureName	Lets you specify the name of the texture.
.TextureStyle	Lets you specify the name of the texture style.
.TextureWidth	Lets you specify the width of the tile in tenths of a micron.
.TextureHeight	Lets you specify the height of the tile in tenths of a micron.
.TextureOffsetX	Lets you specify the amount to offset the texture horizontally.
.TextureOffsetY	Lets you specify the amount to offset the texure vertically.
.RowOffset	Set to TRUE (-1) enables the row offset. Set to FALSE (0) enables the column offset.
.RowColumnOffset	Lets you specify the amount to offset the row or column in tenths of a micron.
.ScaleWithObject	Set to TRUE (-1) transforms the fill when the object is transformed.
.RotationAngle	Lets you specify the amount that the tile is rotated in millionths of degrees.
.SkewAngle	Lets you specify the amount that the tile is skewed in millionths of degrees.
.Starting	Lets you specify the starting transparency.
.Operation	Lets you specify which operation to perform on the selected object.
.Freeze	Set to TRUE (-1) freezes the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyTwoColorBitmapLens (DRAW)

.ApplyTwoColorBitmapLens .FileName = *filename*,.TileWidth = *long*, .TileHeight = *long*, .FirstTileOffsetX = *long*, .FirstTileOffsetY = *long*, .RowOffset = *boolean*, .RowColumnOffset = *long*, .SeamlessTiling = *boolean*, .ScaleWithObject = boolean, .RotationAngle = *long*, .SkewAngle = *long,* .Starting = *long,* .Ending = *long,* .Operation = *long,* .Freeze = *long*

This command applies a two-color bitmap lens to the selected object.

Parameter	Description
.FileName	Lets you specify the path and filenameof the two-color fill file to use.
.TileWidth	Lets you specify the width of the tile in units specified by ScaleWithObject command.
.TileHeight	Lets you specify the height of the tile in units specifies by the ScaleWithObject command.
.FirstTileOffsetX	Lets you specify the amount to offset the first tile horizontally.
.FirstTileOffsetY	Lets you specify the amount to offset the first tile vertically.
.RowOffset	Set to TRUE (-1) enables the row offset. Set to FALSE (0) enables the column offset.
.RowColumnOffset	Lets you specify the amount to offset the row or column. Choose between 0 and 100.
.Seamless Tiling	Set to TRUE (-1) enables seamless tiling.
.ScaleWithObject	Set to TRUE (-1) sets the height/width units to a tenth of a micron. Set to FALSE (0) set the measurement as a percentage.
.RotationAngle	Lets you specify the amount that the tile is rotated in millionths of degrees.
.SkewAngle	Lets you specify the amount that the tile is skewed in millionths of degrees.
.Starting	Lets you specify the starting transparency.
.Operation	Lets you specify which operation to perform on the selected object.
.Freeze	Set to TRUE (-1) freezes the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ApplyUniformBitmapLens (DRAW)

.ApplyUniformBitmapLens .Starting = *long*, .Operation = *long*, .Freeze = *boolean*

This command applies a uniform bitmap lens to the selected object.

Parameter	Description
.Starting	Lets you specify the starting transparency.
.Operation	Lets you specify which operation to perform on the selected object.
.Freeze	Set to TRUE (-1) freezes the object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ClearEffect (DRAW)

.ClearEffect

This command removes a special effect from the selected object.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

ConvertBitmapTo (DRAW)

.ConvertBitmapTo .EffectID = *long*

This command converts the selected bitmap to another bit depth.

Parameter	Description
.EffectID	Lets you specify ID.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ConvertToBitmap (DRAW)

.ConvertToBitmap .BitDepth = *long*, .Grayscale = *boolean*, .Dithered = *boolean*, .TransparentBG = *boolean*, .Resolution = *long*, .AntiAliasing = *long*, .UseColorProfile = *boolean*

This command converts a vector object to a bitmap.

Parameter	Description
.BitDepth	Lets you specify bit depth.
.Grayscale	Set to TRUE (-1) convert to grayscale.
.Dithered	Set to TRUE (-1) enables dithering.
.TransparentBG	Set to TRUE (-1) enables transparent background.
.Resolution	Lets you specify the resolution.
.AntiAliasing	Lets you specify the anti aliasing: 0 = None 1 = Normal 2 = Super Sampling
.UseColorProfile	Set to TRUE (-1) use the color profile.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

CopyPowerClip (DRAW)

.CopyPowerClip .Index = *long*, .XDisp = *long*, .YDisp = *long*

This command copies the PowerClip of the object to the current selection.

Parameter	Description
.Index	Lets you specify the object ID.
.XDisp	Lets you specify the X coordinate of the PowerClip in the object to copy.
.YDisp	Lets you specify the Y coordinate of the PowerClip in the object to copy.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

CopyEffectFrom (DRAW)

.CopyEffectFrom .Clone = *boolean*, .SourceObjectID = *long*

This command copies an effect from a specific object to the selected object.

Parameter	Description
.Clone	Set to TRUE (-1) to clone the effect instead of copying it. This creates a link between the two objects. When the effect is changed for one object, the other object also changes.
.SourceObjectID	Lets you specify the object ID of the source object. Use .GetObjectsCDRStaticID to get an object's ID.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

DetachBlendPath (DRAW)

.DetachBlendPath

This command detaches a path from a blend group.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ExtractContents (DRAW)

.ExtractContents

This command extracts the contents of the selected PowerClip.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

FuseBlend (DRAW)

.FuseBlend .End = *boolean*, .PositionX = *long*, .PositionY = *long*

This command fuses a split blend group.

Parameter	Description
.End	Set to TRUE (-1) to fuse the top of the blend. Set to FALSE (0) to fuse the bottom of the blend.
.PositionX	Lets you specify the X-coordinate of the point where you want the blend to be fused, in tenths of a micron, relative to the center of the page.
.PositionY	Lets you specify the Y-coordinate of the point where you want the blend to be fused, in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

GetBitmapResolution (DRAW)

.GetBitmapResolution .XRes = *long*, .YRes = *long*

This command returns the bitmap resolution.

Parameter	Description
.XRes	Lets you specify the horizontal resolution.
.YRes	Lets you specify the vertical resolution.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

GetBitmapSize (DRAW)

.GetBitmapSize .Width = *long*, .Height = *long*

This command returns the bitmap size.

Parameter	Description
.Width	Lets you specify the bitmap width.
.Height	Lets you specify the bitmap height.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

InflateBitmap (DRAW)

.InflateBitmap .Width = *long*, .Height = *long*, .InflationType = *long*

This command inflates a bitmap.

Parameter	Description
.Width	Lets you specify the bitmap width.
.Height	Lets you specify the bitmap height.
.InflationType	Lets you specify the inflation type: 0 = Size 1 = incrementation 2 = Percentage

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

IsBitmapExternallyLink (DRAW)

.IsBitmapExternallyLink

This command returns true if the bitmap is externally linked.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

PlaceInside (DRAW)

.PlaceInside .Index = *long*, .XDisp = *long*, .YDisp = *long,* .ForceCenter = *boolean,* .Center = *boolean*

This command will PowerClip the current selection.

Parameter	Description
.Index	Lets you specify the object ID.
.XDisp	Lets you specify the X coordinate in the object to PowerClip.
.YDisp	Lets you specify the Y coordinate in the object to PowerClip.
.ForceCenter	Set to TRUE (-1) will use the Center parameter which centers the object inside the container. Set to FALSE (0) Uses the workspace preferences.
.Center	Set to TRUE (-1) center the objet inside the container.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

ResolveAllBitmapsLink (DRAW)

.ResolveAllBitmapsLink

This command resolves all external links.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

ResolveBitmapLink (DRAW)

.ResolveBitmapLink

This command resolves any external links.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

SplitBlend (DRAW)

.SplitBlend .PositionX = *long*, .PositionY = *long*

This command splits the selected blend group.

Parameter	Description
.PositionX	Lets you specify the X-coordinate of the point where you want the blend to be split, in tenths of a micron, relative to the center of the page.
.PositionY	Lets you specify the Y-coordinate of the point where you want the blend to be split, in tenths of a micron, relative to the center of the page.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} **Related Topics**

StartEditContents (DRAW)

.StartEditContents

This command start to edit the contents of the selected PowerClip.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

StopEditContents (DRAW)

.StopEditContents

This command stops to edit the contents of the selected PowerClip.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

UpdateBitmapLink (DRAW)

.UpdateBitmapLink

This command updates any external links.

{button ,AL(`OVR1 Special effects commands;',0,"Defaultoverview",)} Related Topics

Object Data Manager commands

AddObjectDataField (DRAW)

.AddObjectDataField.FieldName = *string*

This function adds an object data field. The new object data field must be followed by its definition. See DefineObjectDataField.

Parameter	Description
.FieldName	Lets you specify the name of object data field.

Example

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

ClearAllObjectData (DRAW)

.ClearAllObjectData

This function clears all the fields and data from the target object.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

ClearObjectData (DRAW)

.ClearObjectData .FieldName = *string*

This function clears a field and data from the selected object.

Parameter	Description
.FieldName	Lets you specify the name of the objects data field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

CopyObjectDataFields (DRAW)

.CopyObjectDataFields.Index = *long*

This function copies all data fields from the target object.

Parameter	Description
.Index	Lets you specify the object ID.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

CreateObjectDataField (DRAW)

.CreateObjectDataField .FieldName = *string,* .Format = *string,* .Order = *long,* .ColumnWidth = *long,* .DrawDefault = *boolean,* .DocDefault = *boolean,* .SummarizeGroup = *boolean*

This function creates a new object data field.

Parameter	Description
.FieldName	Lets you specify the name of object data field.
.Format	Lets you specify the format for the field.
.Order	Lets you specify the position in the field.
.ColumnDefault	Lets you specify the width of the column.
.DrawDefault	Set to TRUE (-1) object is saved in Coreldrw.ini.
.DocDefault	Set to TRUE (-1) object is added to all objects.
.SummarizeGroup	Set to TRUE (-1) to summarize groups.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} **Related Topics**

DefineObjectDataField (DRAW)

.DefineObjectDataField .FieldName = *string,*.Format = *string,*.DrawDefault = *boolean,*.DocDefault = *boolean,*.SummarizeGroup = *boolean*

This function defines the properties of an object data field.

Parameter	Description
.FieldName	Lets you specify the name of object data field.
.Format	Lets you specify the name of the field format.
.DrawDefault	Set to TRUE (-1) object is saved in Coreldrw.ini.
.DocDefault	Set to TRUE (-1) object is added to all objects.
.SummarizeGroup	Set to TRUE (-1) to summarize groups.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} **Related Topics**

DeleteObjectDataField (DRAW)

.DefineObjectDataField .FieldName = *string*

This function deletes the object data field.

Parameter	Description
.FieldName	Lets you specify the name of object data field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

GetObjectData (DRAW)

ReturnString$ = .GetObjectData(.FieldName = *string*)

This function returns a specified object data field of a selected object. If more than one object is selected, the function returns the specified object-data field of the last selected object.

Parameter	Description
ReturnString$	Returns the object data of the selected object.
.FieldName	Lets you specify the name of object data field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} **Related Topics**

GetUserDataField (DRAW)

ReturnString$ = .GetUserDataField(.FieldName = *string*)

This function returns a specified user-data field of a selected object. If more than one object is selected, the function returns the specified user-data field of the last selected object.

Parameter	Description
ReturnString$	Returns the user data field of the selected object.
.FieldName	Lets you specify the name of an object's user data field.

Example

```
u_d_f$="CDRStaticID"
data_field1$=.GetUserDataField (u_d_f)
data_field2$=.GetUserDataField ("Name")
```

The above example returns the value for the CDRStaticID and Name field of a selected object.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

OrderObjectDataFields (DRAW)

.OrderObjectDataFields .FieldToMove = *string,* .NewPosition = *long*

This function changes the order of the object data fields.

Parameter	Description
.FieldToMove	Lets you specify the name of the field to move.
.NewPosition	Lets you specify the position to place the field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

PasteObjectData (DRAW)

.PasteObjectData .Index = *long,* .FieldName = *string*

This function pastes the contents from the Clipboard into a field.

Parameter	Description
.Index	Lets you specify the object ID.
.FieldName	Lets you specify the name of the object data field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

RenameObjectDataField (DRAW)

.RenameObjectDataField .OldFieldName = *string*, .NewFieldName = *string*

This command lets you rename an object data field.

Parameter	Description
.OldFieldName	Lets you specify the old name of the object data field.
.NewFieldName	Lets you specify the new name of the object data field.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

SetObjectData (DRAW)

.SetObjectData .FieldName = *string*, .FieldValue = *string*

This command lets you set object data values for selected objects.

Parameter	Description
.FieldName	Lets you specify the name of the object data field to set.
.FieldValue	Lets you specify the value of the object data field to set.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

SetUserDataField (DRAW)

.SetUserDataField .FieldName = *string*, .FieldValue = *string*

This command lets you set user data field values for selected objects.

Parameter	Description
.FieldName	Lets you specify the name of the user data field to set.
.FieldValue	Lets you specify the value of the user data field to set.

Example

```
.CreateRectangle 1000000, 750000, 500000, 100000, 0
.SetUserDataField "Name", "MyObject"
```

The above example creates a rectangle and while it is still selected, sets its object name to "MyObject". Other common data fields for objects include cost and comments.

{button ,AL(`OVR1 Object Data Manager commands;',0,"Defaultoverview",)} Related Topics

Recorder commands

RecorderBeginEditText (DRAW)

.RecorderBeginEditText

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEndEditText (DRAW)

.RecorderEndEditText

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} Related Topics

RecorderEditTextCharAttributes (DRAW)

.RecorderEditTextCharAttributes .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .FontName = *string*, .FontStyle = *long*, .PointSize = *long*, .Underline = *long*, .Overline = *long*, .StrikeOut = *long*, .Placement = *long*, .CharacterSpacing = *long*, .WordSpacing = *long*, .LineSpacing = *long*, .Alignment = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} Related Topics

RecorderEditTextChangeCase (DRAW)

.RecorderEditTextChangeCase .CaseID = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} Related Topics

RecorderEditReplaceText (DRAW)

.RecorderEditTextReplaceText .NewText = *string*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderBeginEditParaText (DRAW)

.RecorderBeginEditParaText

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEndEditParaText (DRAW)

.RecorderEndEditParaText

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEditParaTextChangeCase (DRAW)

.RecorderEditParaTextChangeCase .CaseID = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEditParaTextSpacing (DRAW)

.RecorderEditParaTextSpacing .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .CharacterSpacing = *long*, .WordSpacing = *long*, .LineSpacing = *long*, .BeforeParagraph = *long*, .AfterParagraph = *long*, .Alignment = *long*, .AutoHyphenation = *boolean*, .HyphenHotZone = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEditParaTextIndents (DRAW)

.RecorderEditParaTextIndents .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .FirstLine = *long*, .RestOfLines = *long*, .RightMargin = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} Related Topics

RecorderEditParaTextCharAttributes (DRAW)

.RecorderEditParaTextCharAttributes .FirstSelectedChar = *long*, .LastSelectedChar = *long*, .FontName = *string*, .FontStyle = *long*, .PointSize = *long*, .Underline = *long*, .Overline = *long*, .StrikeOut = *long*, .Placement = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderEditParaReplaceText (DRAW)

.RecorderEditParaTextReplaceText .NewText = *string*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderApplyPerspective (DRAW)

.RecorderApplyPerspective .Type = *long*, .Flags = *long*, .Box0X = *long*, .Box0Y = *long*, .Box1X = *long*, .Box1Y = *long*, .Box2X = *long*, .Box2Y = *long*, .Box3X = *long*, .Box3Y = *long*, .VPHorizRef = *long*, .VPHorizX = *long*, .VPHorizY = *long*, .VPVertRef = *long*, .VPVertX = *long*, .VPVertY = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderObjectScaleInfo (DRAW)

.RecorderObjectScaleInfo .ScaledSizeX = *long*, .ScaledSizeY = *long*, .DisplacementX = *long*, .DisplacementY = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderSelectObjectByIndex (DRAW)

.RecorderSelectObjectByIndex .ClearFirst = *boolean*, .IIndex = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderSelectObjectsByIndex (DRAW)

.RecorderSelectObjectsByIndex .ClearFirst = *boolean*, .Index1 = *long*, .Index2 = *long*, .Index3 = *long*, .Index4 = *long*, .Index5 = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderSelectPreselectedObjects (DRAW)

.RecorderSelectPreselectedObjects .ClearFirst = *boolean*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

RecorderStorePreselectedObjects (DRAW)

.RecorderStorePreselectedObjects .ConvertingPreset = *boolean*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

StartOfRecording (DRAW)

.StartOfRecording

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

EndOfRecording (DRAW)

.EndOfRecording

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

MenuCommand (DRAW)

.MenuCommand .MenuID = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

ClickedDialogButton (DRAW)

.ClickedDialogButton .DialogID = *long*, .ItemD = *long*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

ShareExtrudeVP (DRAW)

.ShareExtrudeVP .ExtrudeIndex = *long*, .VPToShareIndex = *long*, .SharedVP = *boolean*

This command is used by the Script And Preset Manager. You do not need to include this command in your script.

{button ,AL(`OVR1 Recorder commands;',0,"Defaultoverview",)} **Related Topics**

ID	Color Model	Color 1	Color 2	Color 3	Color 4
1	Pantone	Pantone ID number	Tint (0 - 100)	Ignored	Ignored
2	CMYK100	Cyan (0 - 100)	Magenta (0 - 100)	Yellow (0 - 100)	Black (0 - 100)
3	CMYK255	Cyan (0 - 255)	Magenta (0 - 255)	Yellow (0 - 255)	Black (0 - 255)
4	CMY	Cyan (0 - 255)	Magenta (0 - 255)	Yellow (0 - 255)	Ignored
5	RGB	Red (0 - 255)	Green (0 - 255)	Blue (0 - 255)	Ignored
6	HSB	Hue (0 - 360)	Saturation (0 - 255)	Brightness (0 - 255)	Ignored
7	HLS	Hue (0 - 360)	Lightness (0 - 255)	Saturation (0 - 255)	Ignored
8	Black and White	Black (0) or White (1)	Ignored	Ignored	Ignored
9	Grayscale	Black % (0-255)	Ignored	Ignored	Ignored
10	YIQ255	Y-luminance (0 - 255)	I-chromaticity (0 - 255)	Q-chromaticity (0 - 255)	Ignored
11	L*a*b*	L*-lightness (0 - 255)	a*-green to red (0 - 255)	b*-blue to yellow (0 - 255)	Ignored

ABS function

ABS(x)

Returns the absolute value of a number. Absolute value is the positive value of a number.

Parameter	Description
x	Any numeric expression.

Examples

```
x% = ABS(-3)
y = ABS(3)
```

Both the above examples return 3 to **x** and **y**.

{button ,AL(`abs;sgn;fix;int;;',0,"Defaultoverview",)} Related Topics

ACOS function

ACOS(x)

Returns the inverse cosine (arc cosine) of a given value. The result is an angle measured in radians between 0 and π (where π is approximately 3.14152).

Parameter	Description
x	A numeric expressionbetween -1 and 1.

Note

- To convert the result from radians to degrees, use the **ANGLECONVERT** function or multiply the result by 180/3.14152.

Examples

```
v = ACOS(-0.75)
w = ACOS(0.75)
x = ACOS(0)
y = ACOS(1)
```

In the above example, **v** is equal to 2.418858406, **w** is equal to 0.722734248, **x** is equal to 1.570796327, and **y** is equal to 0.

{button ,AL(`Math_PASTE;ACOS;ASIN;ATAN;;',0,"Defaultoverview",)} Related Topics

ASIN function

ASIN(x)

Returns the inverse sine (arc sine) of a given value. The result is an angle in radians bounded by -p /2 and p /2.

Parameter	Description
x	A numeric expressionbetween -1 and 1.

Note

- To convert the result from radians to degrees, use the **ANGLECONVERT** function or multiply the result by 180/3.14152.

Examples

```
w = ASIN(-0.75)
x = ASIN(0.75)
y = ASIN(0)
z = ASIN(1)
```

In the above example, **w** is equal to -0.8480621, **x** is equal to 0.8480621, **y** is equal to 0, and **z** is equal to 1.570796.

{button ,AL(`Math_PASTE;ACOS;ASIN;ATAN;;',0,"Defaultoverview",)} Related Topics

ATAN function

ATAN(x)

Returns the inverse tangent (arc tangent) of a given value. The result is an angle bounded by -π /2 (-90 degrees) and π /2 (90 degrees) measured in radians.

Parameter	Description
x	A numeric expression.

Note

- To convert the result from radians to degrees, use the **ANGLECONVERT** function or multiply the result by 180/3.14152.

Examples

```
w = ATAN(-0.75)
x = ATAN(0.75)
y = ATAN(0)
z = ATAN(1)
```

In the above example, **w** is equal to -0.6435011, **x** is equal to 0.6435011, **y** is equal to 0, and **z** is equal to 0.7853982.

{button ,AL(`Math_PASTE;ACOS;ASIN;ATAN;;',0,"Defaultoverview",)} Related Topics

COS function

COS(x)

Returns the cosine of an angle measured in radians.

Parameter	Description
x	Any numeric expression. Lets you specify the angle measured in radians.

Notes

- The result of COS is between -1 and 1.
- To convert degrees to radians, multiply degrees by 3.14159/180 (π is approximately equal to 3.14159) or use the **ANGLECONVERT** function.

Examples

```
degreeMeasure% = 45
MyResult = COS(3.14159/180*degreeMeasure%)
```

The above example returns the COS of 45 degrees as expressed in radians. The variable **MyResult** equals 0.707106781.

{button ,AL(`Math_PASTE;CS_MATH_FNS;;;;',0,"Defaultoverview",)} Related Topics

DEC function

DEC(x)

Returns the conversion of a hexadecimal value into decimal notation (as a long data type).

Parameter	Description
x	A string expression representing a hexadecimal number.

Notes

- Decimal notation is a numerical system based on groups of ten units.
- The highest value you can convert is 7FFFFFFF.
- The **HEX** function performs the opposite conversion, from decimal to hexadecimal.

Examples

```
x& = DEC("A27")
```

In the above example, **x** equals 2599.

{button ,AL(`hex;;;;;',0,"Defaultoverview",)} Related Topics

EXP function

EXP(x)

Raises *e* to a given exponent where *e* is the base of the natural logarithm which equals 2.718281828.

Parameter	Description
x	Any numeric expression.

Note

- EXP is the inverse of the natural logarithm (**LN**).

Example

```
x = EXP(7.89)
```

In the above example, **x** equals 2670.444.

{button ,AL(`log;ln;exp;;;',0,"Defaultoverview",)} Related Topics

FIX function

FIX(x)

Removes an argument's decimal or fraction and rounds towards 0. An integer is returned.

Parameter	Description
x	A numeric expression.

Note

- Both **INT** and FIX return the integer portion of a given number. However, INT returns the greatest integer less than or equal to the number, while FIX returns the integer portion given, without any decimal points represented. As a result, -5.26 becomes -5 under the FIX function, and -6 under the INT function.

Examples

```
vv = FIX(12.65)
yy = FIX(-47.29)
```

In the above example, **vv** equals 12 and **yy** equals -47.

```
v = INT(12.65)
y = INT(-47.29)
```

In the above example, **v** equals 12 and **y** equals -48.

{button ,AL(`abs;sgn;fix;int;;',0,"Defaultoverview",)} Related Topics

HEX function

HEX(x)

Converts a number to its corresponding hexadecimal string value.

Parameter	Description
x	A numeric expressionspecifying a decimal value to convert.

Notes

- Decimal numbers are rounded to the nearest whole number before being converted.
- Hexadecimal notation is a numerical system based on groups of sixteen units. Hexadecimal numbers can be expressed in Corel SCRIPT by using the prefix **&h**. For example, &h10 or &h1ABC.
- The **DEC** function performs the opposite conversion, from hexadecimal to decimal.

Example

```
x$ = HEX(27)
x$ = HEX(27.25)
```

In the above example **x** and **y** are both passed the value "1B".

{button ,AL(`dec;;;;;',0,"Defaultoverview",)} Related Topics

INT function

INT(x)

Removes an argument's decimal or fraction and rounds down to the nearest integer. An integer is returned.

Parameter	Description
x	A numeric expression.

Note

- Both INT and **FIX** return the integer portion of a given number. However, INT returns the greatest integer less than or equal to the number, while FIX returns the integer portion given, without any decimal points represented. As a result, -5.26 becomes -5 under the FIX function, and -6 under the INT function.

Examples

```
v = INT(12.65)
y = INT(-47.29)
```

In the above example, **v** equals 12 and **y** equals -48.

```
vv = FIX(12.65)
yy = FIX(-47.29)
```

In the above example, **vv** equals 12 and **yy** equals -47.

{button ,AL(`abs;sgn;fix;int;;',0,"Defaultoverview",)} Related Topics

LN function

LN(x)

Returns the natural logarithm (base *e*) of a number.

Parameter	Description
x	Any positive numeric expression.

Notes

- LN is the inverse of the **EXP** function; therefore LN(EXP(**x**)) equals **x**.
- Natural logarithms are based on the constant *e* which is equal to 2.718281828.

Examples

```
w = LN(1.2)
x = LN(30)
y = LN(1)
z = LN(EXP(30))
```

In the above example, **w** is equal to 0.1823215568, **x** is equal to 3.401197382, **y** is equal to 0, and **z** is equal to 30.

{button ,AL(`log;ln;exp;;;',0,"Defaultoverview",)} Related Topics

LOG function

LOG(x)

Returns the base-10 logarithm of a number.

Parameter	Description
x	Any positive numeric expression.

Note

- The LOG function uses a base of 10. If another base is needed, use LOG(x)/LOG(b) formula where **b** is the base.

Examples

```
x = LOG(25)
y = LOG(5)
```

In the above example, **x** equals 1.397940 and **y** equals 0.6989700.

{button ,AL(`log;ln;exp;;;',0,"Defaultoverview",)} **Related Topics**

RANDOMIZE function

RANDOMIZE x

Sets the random number generator seed to the integer portion of the argument value.

Parameter	Description
x	Any numeric expression. This is an optional parameter. **Randomize** uses the argument as a seed to start a new sequence of random numbers. Using RANDOMIZE without an argument initializes the seed to the system timer. If RANDOMIZE isn't used before calling **RND**, the same sequence of numbers will result every time the random number generator is used.

Examples

```
RANDOMIZE
RANDOMIZE 24
```

In the above example, the first line initializes the random number generator seed to the system timer. The second line uses 24 as the first seed in the random number generator's sequence.

The following example returns 5 random integers between 1 and 10 to the variable **a_random**. Each random value is also displayed in a message box.

```
RANDOMIZE
FOR x = 1 TO 5
   lower=1
   upper=10
   a_random = INT((upper - lower +1)*RND()+lower)
   MESSAGE a_random
NEXT x
```

{button ,AL(`randomize;rnd;;;;',0,"Defaultoverview",)} Related Topics

RND function

RND(x**)**

Returns a random number.

Parameter	Description
x	A numeric expression that determines the bounds of the random number. See the table below.

x's value	Random Number Bounds
Greater than 0	A number between 0 (lower bound) and x (upper bound).
Less than 0	A number between x (lower bound) and 0 (upper bound).
Omitted	A number between 0 and 1.

Note

- If **RANDOMIZE** isn't used before calling **RND** for the first time, the same sequence of numbers will result every time the script is run.

Examples

```
x = RND()
y = RND(-7)
z = RND(24)
```

In the above example, **x** returns a random number between 0 and 1, **y** returns a random number between -7 and 0, and **z** returns a random number between 0 and 24.

To create a random integer in a specified range, use the following formula:

```
INT((upper - lower + 1)*RND()+ lower)
```

where **upper** is the highest number in the specified range and **lower** is the lowest number in the specified range. The following example returns a random integer between 1 and 10 to the variable **a_random**.

```
lower = 1
upper = 10
a_random = INT((upper - lower +1)*RND()+lower)
```

{button ,AL(`randomize;rnd;;;;',0,"Defaultoverview",)} **Related Topics**

SGN function

SGN(x)

Determines the sign (+ or -) of a number. Returns -1 if the number is negative, 1 if the number is positive, and 0 if the number is 0.

Parameter	Description
x	Any numeric expression.

Examples

```
x% = SGN(5)
y = SGN(0)
z = SGN(-0.021)
```

In the above example, **x** is equal to 1, **y** is equal to 0, and **z** is equal to -1.

{button ,AL(`abs;sgn;fix;int;;',0,"Defaultoverview",)} Related Topics

SIN function

SIN(x)

Returns the sine of an angle measured in radians.

Parameter	Description
x	Any numeric expression. Lets you specify the angle measured in radians.

Note

- The result of SIN is between -1 to 1, inclusive.
- To convert degrees to radians, multiply degrees by 3.14159/180 (π is approximately equal to 3.14159) or use the **ANGLECONVERT** function.

Examples

```
degreeMeasure% = 45
MyResult = SIN(3.14159/180*degreeMeasure%)
```

The above example returns the SIN of 45 degrees. The variable **MyResult** equals 0.70710678.

{button ,AL(`Math_PASTE;CS_MATH_FNS;;;;',0,"Defaultoverview",)} Related Topics

SQR function

SQR(x)

Returns the positive square root of a number.

Parameter	Description
x	Any non-negative numeric expression.

Examples

```
w = SQR(4)
x = SQR(0.175)
```

In this example, **w** is equal to 2 and **x** is equal to 0.4183300133.

{button ,AL(`abs;sgn;fix;int;;',0,"Defaultoverview",)} Related Topics

TAN function

TAN(x)

Returns the tangent of an angle measured in radians.

Parameter	Description
x	Any numeric expression. Lets you specify the angle measured in radians.

Note

- To convert degrees to radians, multiply degrees by 3.14159/180 (*p* • is approximately equal to 3.14159) or use the **ANGLECONVERT** function.

Example

```
radianMeasure = 0.5
MyResult = TAN(radianMeasure)
```

The above example returns the TAN of 0.5 radians. The variable **MyResult** equals 0.546302.

{button ,AL(`Math_PASTE;CS_MATH_FNS;;;;',0,"Defaultoverview",)} Related Topics

GETCOLOR statement and function

Statement syntax
GETCOLOR Red, Green, Blue

Function syntax
ReturnValue = GETCOLOR (Red, Green, Blue)

Displays a standard Windows Color dialog box and returns color setting values from the RGB color model (Red, Green, Blue).

Return Value

The GETCOLOR function returns one of the following values

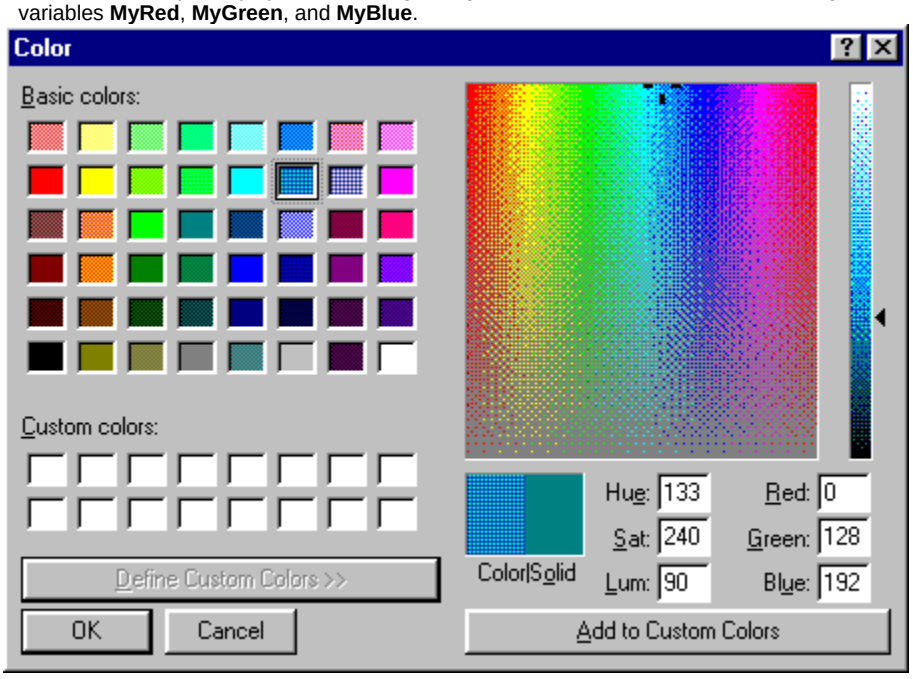
- TRUE (-1)▶the Cancel button was not clicked
- FALSE (0)▶the Cancel button was clicked

Parameter	Definition
Red	Lets you specify the numeric variable that is passed the Red setting of the selected color (0 - 255). You can also use this variable to set an initial value.
Green	Lets you specify the numeric variable that is passed the Green setting of the selected color (0 - 255). You can also use this variable to set an initial value.
Blue	Lets you specify the numeric variable that is passed the Blue setting of the selected color (0 - 255). You can also use this variable to set an initial value.

Example

GETCOLOR MyRed%, MyGreen%, MyBlue%

The above example displays the following dialog box and returns the RGB color settings for the selected color to the numeric variables **MyRed**, **MyGreen**, and **MyBlue**.



{button ,AL(`cs_ui_statements;;;;;',0,"Defaultoverview",)} **Related Topics**

GETFILEBOX function

GETFILEBOX(Filter, Title, Type, DefFile, DefExt, DefFol, BtnName**)**

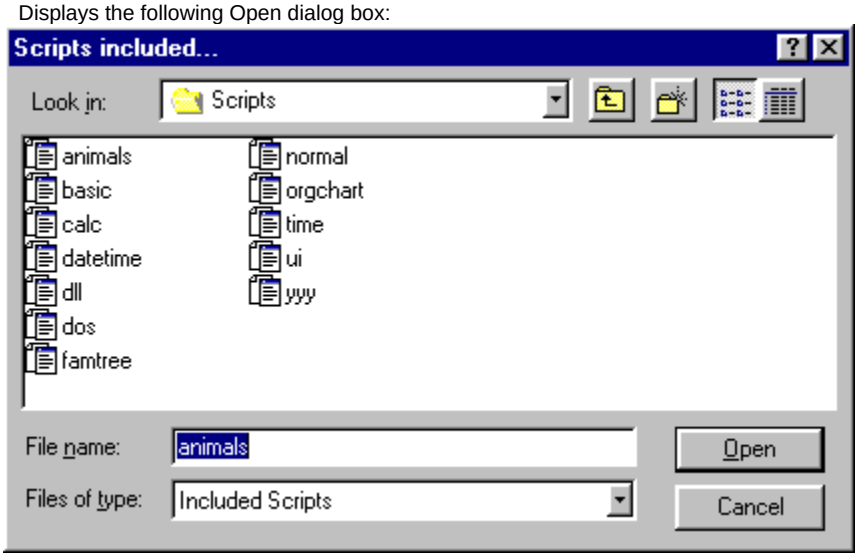
This function displays a standard Windows File Open or File Save As dialog box. Both dialog boxes allow users to choose a file from the file system. The **GETFILEBOX** function returns the selected filename and its full path, or an empty string if the user chooses Cancel. The **GETFILEBOX** statement by itself does not open or save a file; it only returns a string corresponding to the selected file.

Parameter	Definition
Filter	String expression specifying the filters to use in the dialog box. For the Open dialog box, the filters are listed in the Files of Type list box. For the Save As dialog box, the filters are listed in the Save as Type list box. Filters are specified in two parts. The first part is the text that appears in the list box, and the second part is the actual filter extension. The parts are separated by the | character (do not use spaces before or after the | characters). To separate multiple filters, use the | character. See the example below for more information.
Title	String expression specifying the title to display in the dialog box. If not specified, "Open" is displayed for an Open dialog box and "Save As" is displayed for a Save As dialog box.
Type	Numeric expression specifying the type of dialog box to display: 0 File Open dialog box (default if omitted) 1 File Save dialog box
DefFile	String expression specifying the text to display in the File name text box of the dialog box. If not specified, the text box is empty.
DefExt	String expression specifying the default extension to append to a File name if the user omits the extension.
DefFol	String expression specifying the default folder used by the dialog box. If not specified, or the specified folder does not exist, the current folder is used.
BtnName	String expression specifying a button name to override the Open or Save button in the dialog box. If not specified, the button's name remains unchanged.

Example

```
SETCURRFOLDER = "c:\COREL50\DRAW\samples" 'set the current folder
Filename$=GETFILEBOX("Included Scripts|*.csc|All Files|*.*", "Scripts included...", 0,"animals")
```

Displays the following Open dialog box:



{button ,AL(`cs_ui_statements;chfolder;;;;',0,"Defaultoverview",)} **Related Topics**

GETFOLDER function

GETFOLDER (InitFolder**)**

This function displays a Windows Choose Folder dialog box. The Choose Folder dialog box returns the folder and path a user chooses as a string.

Parameter	Definition
InitFolder	String expression specifying the default path and folder to display in the dialog box. If not specified, the active folder is used.

Note

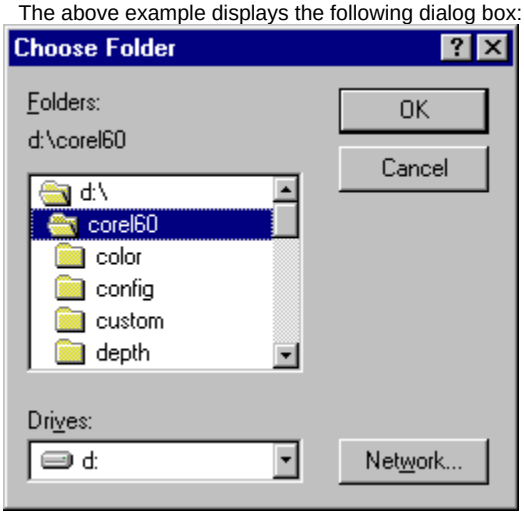
- If the Cancel button is clicked, an empty string is returned.

Example

NewFolder$ = GETFOLDER("D:\Corel60")

SETCURRFOLDER = NewFolder$

The above example displays the following dialog box:



The selected folder is passed to the string variable **NewFolder**. The **SETCURRFOLDER** statement sets the current folder to the folder name passed to **NewFolder**.

{button ,AL(`cs_ui_statements;chfolder;currfolder;;;',0,"Defaultoverview",)} Related Topics

GETFONT statement and function

Statement syntax
GETFONT FaceName, PointSize, Weight, Italic, Underline, StrikeOut, Red, Green, Blue

Function syntax
ReturnValue = GETFONT (FaceName, PointSize, Weight, Italic, Underline, StrikeOut, Red, Green, Blue)

Displays a standard Windows Font dialog box and returns the selected font settings.

Return Value

The GET font function returns one of the following values:

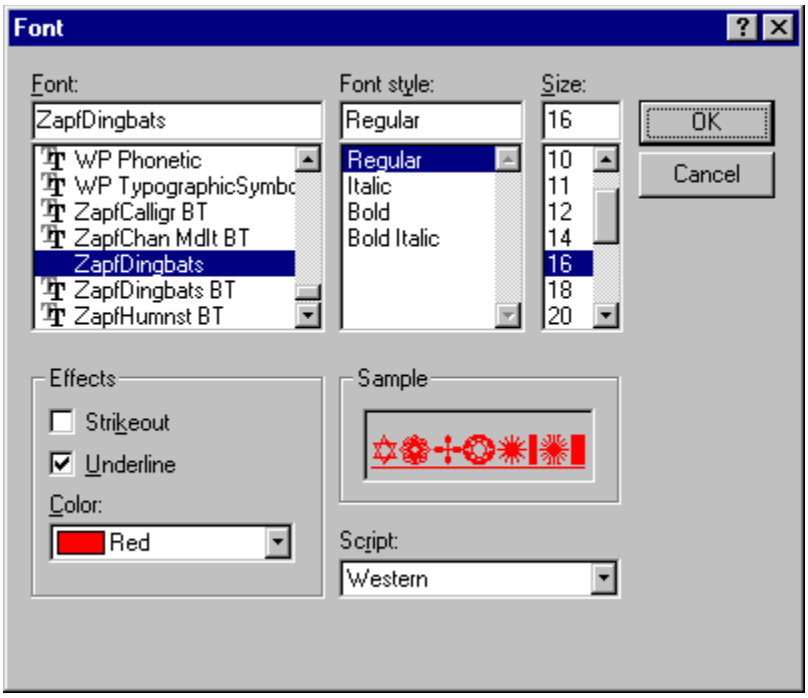
- TRUE (-1) ▶ the Cancel button was not clicked
- FALSE (0) ▶ the Cancel button was clicked

Parameter	Definition
FaceName	Lets you specify a string variable that is passed the name of the selected font. You can also use this variable to set an initial value.
PointSize	Lets you specify a numeric variable that is passed the font size in points. You can also use this variable to set an initial value. This parameter uses non-fractional values
Weight	Lets you specify a numeric variable that is passed the font's weight setting (number of inked pixels per 1000 pixels). Common values and their corresponding names include: 100 Thin 200 Extra Light, Ultra Light 300 Light **400 Normal, Regular** 500 Medium 600 Semi Bold, Demi Bold **700 Bold** 800 Extra Bold, Ultra Bold 900 Black, Heavy Most Windows fonts only use two weight settings: 400 (Normal) and 700 (Bold).
Italic	Lets you specify a numeric variable that is passed the font's italic setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
Underline	Lets you specify a numeric variable that is passed the font's underline setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
StrikeOut	Lets you specify a numeric variable that is passed the font's strike out setting: TRUE (-1) if this setting is enabled; FALSE (0) otherwise.
Red	Lets you specify the numeric variable that is passed the Red (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.
Green	Lets you specify the numeric variable that is passed the Green (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.
Blue	Lets you specify the numeric variable that is passed the Blue (RGB color model) setting of the selected font's color (0 - 255). You can also use this variable to set an initial value.

Example

GETFONT FN, PS, Wt, Italic, UL, SO, R, G, B

The above example displays the following dialog box and returns the font settings the variables specified.



{button ,AL(`cs_ui_statements;;;;;',0,"Defaultoverview",)} **Related Topics**

INPUTBOX function

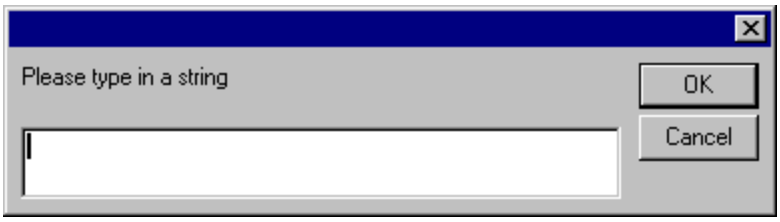
INPUTBOX(prompt$)

Displays a simple dialog box where you can enter a string that is returned to a script. The dialog box has OK and Cancel buttons. If the Cancel button is chosen, an empty string is returned.

Parameter	Definition
prompt$	String expression that appears in the dialog box above the edit box.

Example

MyString$ = INPUTBOX("Please type in a string")



User input is returned to the variable **MyString$.**

{button ,AL(`cs_ui_statements;textbox;text;;;',0,"Defaultoverview",)} Related Topics

MESSAGE statement

Message anyVariable

Displays a dialog box that contains a specified message and an OK button. This statement doesn't return any value to a running script, but can provide the script user with information during script execution.

Parameter	Definition
anyVariable	Any numeric or string expression to display in the message box. Numbers and dates are displayed as their string representations.

Example

```
x$="Hello." + CHR(13)     'CHR(13) is a return character
MESSAGE x$ + "What a nice day."
```

Note

- See the Corel SCRIPT character map for a list of character codes.

{button ,AL(`cs_ui_statements;textbox;text;chr;;',0,"Defaultoverview",)} Related Topics

MESSAGEBOX function

MESSAGEBOX(prompt, title, option**)**

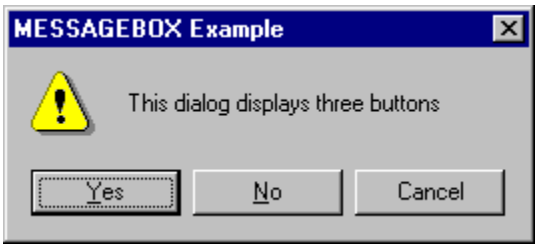
Displays a message box with a specified message and user-specified buttons and icons. MESSAGEBOX returns a value representing the button that was used to close it to the script.

Parameter	Definition
prompt	String expression to display in the box.
title	String expression to display in the message box caption.
option	A numeric expression representing the type of buttons to include in the box and an icon (if any) to appear beside the message. The **option** value is set using the OR (or +) operator for multiple buttons (see example): **Button Type:** 0 OK only (used by default if a button is not specified) 1 OK/Cancel 2 Abort/Retry/Ignore 3 Yes/No/Cancel 4 Yes/No 5 Retry/Cancel **Icon Type** (click hot spot for an example)**:** 0 No icon 16 Stop 32 Question 48 Exclamation 64 Information

Returns	Button pressed
1	OK
2	Cancel
3	Abort
4	Retry
5	Ignore
6	Yes
7	No

Example

retval% = MESSAGEBOX("This dialog displays three buttons", "MESSAGEBOX Example", 3 OR 48)



Note

- Pressing the Close Dialog button () is the same as pressing the Cancel button; both return 2.

{button ,AL(`cs_ui_statements;textbox;text;;;',0,"Defaultoverview",)} **Related Topics**

BUILDDATE function

BUILDDATE (Year, Month, Day)

Assigns a date value to a date variable.

Parameter	Description
Year	Numeric expression specifying the year to assign to a date variable.
Month	Numeric expression specifying the month to assign to a date variable. Valid values are from 1 to 12 inclusive.
Day	Numeric expression specifying the day to assign to a date variable. Valid values are from 1 to 31 inclusive, depending on the **Month** setting.

Note

- **BUILDATE** can only accept a date value between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050). If a date outside this range is specified, an error occurs.

Example

```
DIM BigDay AS DATE
BigDay = BUILDDATE(1996, 1, 14)
```

In the above example, the first line declares the date variable **BigDay**. This variable is then assigned the date January 14, 1996.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

BUILDTIME function

BUILDTIME (Hour, Minute, Second)

Assigns a time value to a date variable.

Parameter	Description
Hour	Numeric expression specifying the hour to assign to a date variable. This value is based on a 24-hour clock. For example, 5 PM equals 17. Valid values are from 0 to 23 inclusive.
Minute	Numeric expression specifying the minute to assign to a date variable. Valid values are from 0 to 59 inclusive.
Second	Numeric expression specifying the second to assign to a date variable. Valid values are from 0 to 59 inclusive.

Example

```
DIM BigDay AS DATE
BigDay = BUILDTIME(14, 30, 0)
```

In the above example, the first line declares the date variable **BigDay**. This variable is then assigned the time 2:30 PM.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

FORMATDATE function

DateString = FORMATDATE (DateExp, FormatString)

Converts a date expression into a string with a specified format.

Return Value

Lets you specify the string variable that is assigned the formatted date.

Parameter	Description
DateExp	Lets you specify the date expression to convert to a string.
FormatString	Lets you specify a code, as a string, representing the format of the date. Formats are created by using and combining the following codes.

To format	Use this format code (case-sensitive)
Days as 1-31	d
Days as 01-31	dd
Days as Sun-Sat	ddd
Days as Sunday-Saturday	dddd
Months as 1-12	M
Months as 01-12	MM
Months as Jan-Dec	MMM
Months as January-December	MMMM
Year as 6 in 1996	y
Year as 96 in 1996	yy
Year as 1996	yyy

Note

- You can insert spaces and punctuation between date elements within the formatting string. See the example below.

Example

```
DIM TodayDate AS DATE
TodayDate = GETCURRDATE()
StringDate$ = FORMATDATE (TodayDate, "dddd, MMMM d, yyy")
MESSAGE StringDate
```

In the above example, the first line declares the date variable **TodayDate**. This variable is then assigned the current date with the **GETCURRDATE** function. The **StringDate** variable is then assigned today's date using the formatting shown in the following example.

Saturday, September 16, 1995

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} **Related Topics**

FORMATTIME function

TimeString = FORMATTIME (TimeExp, FormatString)

Converts a time expression into a string with a specified format.

Return Value

Lets you specify the string variable that is assigned the formatted time.

Parameter	Description
TimeExp	Lets you specify the time expression to convert to a string.
FormatString	Lets you specify a code, as a string, representing the format of the time. Formats are created by using and combining the following codes.

To format	Use this format code (case-sensitive)
Hours as 1-12 (12-hour clock)	h
Hours as 01-12 (12-hour clock)	hh
Hours as 0-23 (24-hour clock)	H
Hours as 00-23 (24-hour clock)	HH
Minutes as 0-59	m
Minutes as 00-59	mm
Seconds as 0-59	s
Seconds as 00-59	ss
AM/PM as A or P	t
AM/PM as AM or PM	tt
Time as 4:36 pm	h:mm pm

Note

- You can insert spaces and punctuation between time elements within the formatting string. See the example below.

Example

```
DIM TimeNow AS DATE
TimeNow = GETCURRDATE()
StringTime = FORMATTIME (TimeNow, "HH:mm:ss tt")
MESSAGE StringTime
```

In the above example, the first line declares the date variable **TodayDate**. This variable is then assigned the current date and time with the **GETCURRDATE** function. The **StringTime** variable is then assigned the current time using the formatting shown in the following example.

```
13:46:25 PM
```

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

GETCURRDATE function

DateTime = GETCURRDATE ( )

Returns the system's current date and time.

Return Value

Lets you specify the date variable that is assigned the current date and time. See Assigning values to date variables for more information.

Note

- Formatting used to display dates and time is set in the Windows Control Panel. In Windows 95, see Regional settings for formatting information; in Windows NT, see International settings.
- **GETCURRDATE** can return a date between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050).
- In Corel SCRIPT version 7.0, the **GETCURRDATE** function and the **SETCURRDATE** statement replace the **CURRDATE** statement.

Example

```
DIM DT AS DATE
DT = GETCURRDATE()
MESSAGE DT
```

The above example displays the system date and time in a message box. You can also extract portions of the date and convert them to strings and numbers. The following continues the above example:

```
MyDateString$ = DT                ' create a string
MyDay% = VAL(LEFT(MyDateString$, 2)) ' extract the day as an integer
```

In the above example, the **VAL** and **LEFT** functions are used to extract an integer from the **MyDateString** string variable (created by converting a date variable). The method you use to extract portions from a date string depend on your Windows date settings. You can also use the **GETDATEINFO** or the **GETTIMEINFO** function to extract information from a date variable.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

GETDATEINFO function

GETDATEINFO DateExp, Year, Month, Day, DayOfWeek

Extracts the components of a date expression to numeric variables.

Parameter	Description
DateExp	Lets you specify the date expression to extract components from.
Year	Lets you specify the numeric variable that is assigned the year component from the specified date expression.
Month	Lets you specify the numeric variable that is assigned the month component from the specified date expression.
Day	Lets you specify the numeric variable that is assigned the day component from the specified date expression.
DayOfWeek	Lets you specify the numeric variable that is assigned the day of week component from the specified date expression. Sunday corresponds to 1, Monday to 2, and so on.

Note

- **GETDATEINFO** can only accept a date value between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050). If a date outside this range is specified, an error occurs.

Example

```
DIM TodayDate AS DATE

TodayDate = GETCURRDATE()

GETDATEINFO TodayDate, Y&, M&, D&, DW&
```

In the above example, the first line declares the date variable **TodayDate**. This variable is then assigned the current date with the **GETCURRDATE** function. The variables **Y**, **M**, **D**, and **DW** are then assigned their respective component of the date stored in **TodayDate**. If **TodayDate** was set to May 29, 1996, then **Y**=1996, **M**=5, **D**=29, and **DW**=4.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} **Related Topics**

GETTIMEINFO function

GETTIMEINFO TimeExp, Hour, Minute, Second

Extracts the components of a time expression to numeric variables.

Parameter	Description
TimeExp	Lets you specify the time expression to extract components from.
Hour	Lets you specify the numeric variable that is assigned the hour component from the specified time expression. The number assigned is based on a 24-hour clock. For example, 16 is the numeric variable for 4pm.
Minute	Lets you specify the numeric variable that is assigned the minute component from the specified time expression.
Second	Lets you specify the numeric variable that is assigned the second component from the specified time expression.

Example

```
DIM TodayTime AS DATE

TodayTime = GETCURRDATE()

GETTIMEINFO TodayTime, H&, M&, S&
```

In the above example, the first line declares the date variable **TodayTime**. This variable is then assigned the current date and time with the **GETCURRDATE** function. The variables **H**, **M**, and **S** are then assigned their respective component of the time stored in **TodayTime**. If **TodayTime** was set to 5:37:16 PM, then **H**=17, **M**=37, **S**=16.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} **Related Topics**

Example

SETCURRDATE statement

SETCURRDATE DT

Sets the system's date and time. If used improperly, this statement can cause problems in the system's Windows settings.

Parameter	Description
DT	Lets you specify a date expression that sets the system's date and time

Note

- **SETCURRDATE** can assign a date between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050).
- Formatting used to display dates and time is set in the Windows Control Panel. In Windows 95, see Regional settings for formatting information; in Windows NT, see International settings.
- In Corel SCRIPT version 7.0, the **SETCURRDATE** statement and the **GETCURRDATE** function replace the **CURRDATE** statement.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

Examples for SETCURRDATE

To change the system date and time

The following example sets the system date to August 25, 1995, and the time to 12:00 A.M. A message box is used to display the date value of **dateOnly**.

```
DIM dateOnly AS DATE
dateOnly = 34936
SETCURRDATE dateOnly
MESSAGE dateOnly
```

The following example sets the system date to August 25, 1995, and the system time to 12:05.46 P.M. A message box is used to display the date value of **dateAndTime.**

```
DIM dateAndTime AS DATE
dateAndTime = 34936.504
CURRDATE = dateAndTime
MESSAGE dateAndTime
```

Using SETCURRDATE for file date stamping

The following example assigns the system date to the variable **xDate**. The third line sets the system date to February 26, 1994, and the fourth line opens a text file that is stamped with the new system date. The last line resets the system date to its original value.

```
DIM xDate AS DATE
xDate = GETCURRDATE      'assigns system date
SETCURRDATE "02/26/1994"     'sets the system date
OPEN "c:\log.txt" FOR OUTPUT AS 2
SETCURRDATE xDate
```

Note

- Formatting used to display dates and time is set in the Windows Control Panel. In Windows 95, see Regional settings for formatting information; in Windows NT, see International settings.

{button ,AL(`mid;left;val;cs_date_time;;',0,"Defaultoverview",)} Related Topics

WAIT FOR statement

WAIT FOR x

Pauses script execution for a specified number of seconds.

Argument	Description
x	A non-negative numeric expression specifying the number of seconds to pause script execution.

Example

MESSAGE "Start"

WAIT FOR 3

MESSAGE "Done"

The above example displays a message box, once the message box is closed, script execution pauses for three seconds, and then displays another message box.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} Related Topics

WAIT UNTIL statement

WAIT UNTIL x

Pauses script execution until the system timer matches a specified date serial number.

You should consider using the **WAIT UNTIL** statement to run scripts in off-peak times. For example, if you have a large print job, you could use a script to run it during the night.

Argument	Description
x	Lets you specify the date and time when to resume execution. This parameter must be positive and greater than the system's current date.

Note

- **WAIT UNITL** can be set to a date between January 1, 1980 (date serial number 29221) and December 31, 2099 (date serial number 73050). If a date outside this range is specified, an error occurs.

Example

The following example pauses script execution until the system time matches 34936.25 (August 25, 1996, 6:00 A.M.).

```
DIM offPeak AS DATE
offPeak = 34936.25
WAIT UNTIL offPeak
```

The following example pauses script execution until 3:00 A.M. regardless of the date:

```
DIM today, tonite   AS DATE
DIM daypart AS LONG
today = CURRDATE
daypart = INT(today)   'daypart is set to 12:00 A.M. today
tonite = daypart + 1.125 'sets the date and time to 3 AM the next day
WAIT UNTIL tonite
```

If you add a day and 3 hours (1.125) to **datepart**, **tonite** is set to 3 A.M. the next morning.

{button ,AL(`cs_date_time;;;;;',0,"Defaultoverview",)} **Related Topics**

ADDFOL statement

#ADDFOL folder

This statement adds a temporary folder to the paths Corel SCRIPT searches when trying to find an **INCLUDE** file on your system.

Parameter	Description
folder	String expression specifying a drive and a folder.

Note

- The pound sign (#) is required in the syntax.
- Corel SCRIPT searches for an INCLUDE file in the following order:

1. The folder where the script resides. You can use the **GETSCRIPTFOLDER** statement to set or determine the active folder.
2. Folders in the path. The path is specified in the systems AUTOEXEC.BAT file.
3. Folders set in the Corel SCRIPT Editors **INCLUDE** option. Click  for more information about setting INCLUDE folders.
4. Folders specified with the ADDFOL statement.

Example

```
#ADDFOL "C:\MyFiles"
```

The above example add the MYFILES folder on the C drive to the path Corel SCRIPT searches for INCLUDE files.

{button ,AL(`include;setcurrfolder;;;;',0,"Defaultoverview",)} Related Topics

ADDRESBMP statement

#ADDRESBMP name file

This statement embeds a Windows bitmap graphic (.BMP) into an executable (.EXE), DLL, or Corel Add-on (.CAO) created with Corel SCRIPT or a Corel SCRIPT Binary file (.CSB). If you're distributing a script that uses many bitmaps in the form of an executable, DLL, or Corel SCRIPT Binary file, this statement can reduce the number of files that you must distribute.

In Corel SCRIPT, bitmaps are most often used in custom dialog boxes. See the Image dialog box control for an example of a script using bitmaps.

Parameter	Description
name	String expression specifying the bitmap reference name within a script. When this name is referenced within a script, it must be preceded with the pound sign (#) and be enclosed in quotations. However, within the **ADDRESBMP** statement, quotations should not be used. See the example below for more information.
file	String expression specifying the filename and path of a Windows bitmap graphic.

Note

- The pound sign (#) is required in the syntax.

Example

```
#ADDRESBMP Wizard1Portrait "C:\MyScripts\portrait.bmp"
BEGIN DIALOG Dialog1 200, 100, "Corel SCRIPT Dialog"
    IMAGE  11, 17, 74, 65, "#Wizard1Portrait"
END DIALOG
```

In the above example, the PORTRAIT.BMP bitmap file takes on the reference name Wizard1Portrait. This bitmap is then used in an image control in a custom dialog box. If this script was compiled into an executable the PORTRAIT bitmap would be embedded into the executable file.

{button ,AL(`image;image_dyn;Creating Corel SCRIPT Executables;Distributing_cs_exe;;',0,"Defaultoverview",)}
Related Topics

BEEP statement

BEEP

Sounds a tone.

Note

- The sound your computer makes depends on your computer's hardware (for example, sound cards, PC speaker, and so on) and the Default sound in your Windows sound settings (see your Windows Control Panel for more details).

Example

```
IF (abc<=15.3) then BEEP ELSE MESSAGE "It's greater than 15.3"
```

If the variable **abc** is less than or equal to 15.3, the computer sounds a tone.

{button ,AL(`csui_statements;;;;;',0,"Defaultoverview",)} **Related Topics**

BEGINWAITCURSOR and ENDWAITCURSOR statements

BEGINWAITCURSOR
ENDWAITCURSOR

The **BEGINWAITCURSOR** statement sets the mouse pointer to Busy. The Busy pointer usually appears as an hour-glass on most systems. This command is useful for scripts that perform long operations. By setting the pointer to Busy, the user is alerted that the script is still executing.

The **ENDWAITCURSOR** statement returns the pointer to its normal state. If the **ENDWAITCURSOR** statement is not in a script that uses the **BEGINWAITCURSOR** statement, the pointer reverts to its normal state after the script finishes executing.

Note

- It is good programming practice to use an **ENDWAITCURSOR** statement with every **BEGINWAITCURSOR** statement.

Example

```
BEGINWAITCURSOR
WAIT FOR 30
ENDWAITCURSOR
```

In the above example, the pointer is set to Busy, script execution is paused for 30 seconds, and then the pointer reverts to its normal state.

{button ,AL(`csui_statements;;;;;',0,"Defaultoverview",)} Related Topics

COPY statement and function

COPY file1, file2, overwrite

The **COPY** statement copies a file.

You can also use **COPY** as a function: it returns TRUE (-1) if the **COPY** operation is successful; **FALSE** (0) otherwise.

Parameter	Description
file1	String expression specifying the file to copy. **file1** can include the drive and folder.
file2	String expression specifying the where **file1** is to be copied. **file2** can include the drive and folder.
overwrite	If **file2** already exists, this determines whether to overwrite the existing file: 0 copy and overwrite (default if omitted) 1 overwrite fails

Example

```
DIM x AS STRING
DIM y AS STRING
x = "C:\work\example1.cdr"
y = "D:\work\example1.cdr"
COPY x, y, 0
```

The above example copies the EXAMPLE1.CDR file to the work folder on the D drive.

```
DIM x AS STRING
DIM y AS STRING
x = "C:\work\example1.cdr"
y = "D:\work\example1.cdr"
success = COPY (x, y, 0)
```

The above example copies the EXAMPLE1.CDR file to the Work folder on the D drive, and assigns -1 to **success**.

{button ,AL(`FINDFIRSTFOLDER_FINDNEXTFOLDER;rename;copy;SETCURRFOLDER;rmfolder;GETcurrfolder;;',0,"Defaultoverview",)} Related Topics

FILEATTR function

retval = FILEATTR(FileName)

This functions returns a file's or folder's attributes.

Return Value

The FILEATTR function returns one of the following values:

- 0 file doesn't exist
- 1 read-only files or folder
- 2 hidden files or folder
- 4 system files or folder used by operating system
- 16 folder
- 32 archive files or folder
- 128 normal files or folder
- 256 temporary files or folder
- 2048 compressed files and folders

Parameter	Description
FileName	String expression specifying the file/folder for which the attributes are returned.

Example

```
retval = FILEATTR("C:\myfiles\mysetup.txt")
```

If MYSETUP.TXT is read-only, hidden, and a system file, **retval** equals 7.

In cases where multiple attributes are returned, you can use the AND (bitwise) operator to determine specific attributes. To determine if MYSETUP.TXT was a read-only file you could use the following syntax:

```
IF 1 AND retval THEN readOnly$ = "Yes" ELSE readOnly$ = "No"
```

1 is the read-only attribute. The variable **readOnly** is assigned a string based on bitwise comparison. In this case **readOnly** is assigned "Yes".

{button ,AL(`GETCURRFOLDER;filemode;getfileattr;FINDFIRSTFOLDER_FINDNEXTFOLDER;setcurrfolder;',0,"Defaultoverview",)} Related Topics

FILEDATE function

retval = FILEDATE(FileName)

This functions returns a file's last modification date.

Return Value

Assigned the date of the specified file's last modification as date data type. If the file is not found, 0 is returned.

Parameter	Description
FileName	Lets you specify the file for which the date property is returned.

Note

- If the file is not closed when the function is executed, the function returns 12:00 AM.

Example

```
retval = FILEDATE("C:\myfiles\mytext.txt")
```

{button ,AL(`GETCURRFOLDER;filemode;getfileattr;FINDFIRSTFOLDER_FINDNEXTFOLDER;setcurrfolder;',0,"Defaultoverview",)} Related Topics

FILEMODE function

FILEMODE(num)

Returns the file mode of an open text file in your computer's memory.

Parameter	Description
num	Numeric expression specifying an open file number to examine: 1 file was opened for input 2 file was opened for output 8 file was opened for append

Note

- This the only Corel SCRIPT file function that doesn't have an optional # sign in front of the file number.

Examples

```
i% = FILEMODE(1)
```

If file 1 was opened for input, then **i%** is set to 1. If file 1 was opened for output, then **i%** is set to 2. If file 1 was opened for append, then **i%** is set to 8.

```
OPEN "C:\example.txt" FOR APPEND AS 2
i% = FILEMODE(2)
```

Assigns 8 to the variable **i%**.

{button ,AL(`OPEN_APPEND;OPEN_INPUT;OPEN_OUTPUT;;;',0,"Defaultoverview",)} Related Topics

FILEPOS function

FILEPOS (#num)

Returns the current file position of the file pointer for the specified file.

Parameter	Description
#**num**	Numeric expression specifying an open file number to examine. The # sign is optional.

Note

- The seek always starts from the first position in the file.

Example

```
OPEN "C:\HELLO.TXT" FOR INPUT AS 2
SEEK 2, 12
position% = FILEPOS(2)
```

Assigns 12 to the variable **i%**.

{button ,AL(`lof;seek;open_append;print;write;',0,"Defaultoverview",)} Related Topics

FILESIZE function

retval = FILESIZE(FileName)

Use the FILESIZE function to return a file's size in bytes.

Return Value

Assigned the size of the specified file in bytes. If the file is not found, 0 is returned.

Parameter	Description
FileName	Lets you specify the file for which the date property is returned.

Example

```
retval = FILESIZE("C:\myfiles\mytext.txt")
```

{button ,AL(`GETCURRFOLDER;filemode;getfileattr;FINDFIRSTFOLDER_FINDNEXTFOLDER;setcurrfolder;',0,"Defaultoverview",)} Related Topics

FINDFIRSTFOLDER, FINDNEXTFOLDER functions

FolderFileName$ = FINDFIRSTFOLDER(searchcriteria, attributes)
FolderFileName$ = FINDNEXTFOLDER( )

Use the **FINDFIRSTFOLDER** and **FINDNEXTFOLDER** functions to assemble or perform an operation on a list of files, folders, or both. The **FINDFIRSTFOLDER** function is used to locate the first file or first folder in a folder that meets a specified search criteria. The **FINDNEXTFOLDER** function is used to locate the next file or next folder that meets the specified search criteria set by the **FINDFIRSTFOLDER**. The **FINDNEXTFOLDER** function must be used in conjunction with the **FINDFIRSTFOLDER** function.

Return Value

String variable that is passed the name of the folder.

Parameter	Description
searchcriteria	Lets you specify the files or folders for which to search. You can include wild-card characters (* or ?).
attributes	The type of files or folders you want to use. Use the OR operator to specify multiple file and folder types: 1 read-only 2 hidden 4 system 16 Lets you specify to use folders. If not specified, files are used. 32 archive 128 normal (not read-only, hidden, system or archive file or folder) 256 temporary 2048 compressed

Note

- Specifying folder in the **attributes** parameter (16) by itself does not specify a type of folder. You must use another parameter along with 16 to specify a folder type.

Example

```
DIM DCOUNT%,  FCOUNT%             'creates 2 integer variables
DIM FILESARR$(100), DIRARR$(100)  'creates 2 string arrays

REM LOOP #1
REM FIND ALL DIRECTORIES IN THE SAMPLES FOLDER
DCOUNT = 1
DIRARR(DCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\*", 16 OR 128)
WHILE (DIRARR(DCOUNT) <> "")
    MESSAGE DIRARR(DCOUNT)
    IF DIRARR(DCOUNT) <> "." AND DIRARR(DCOUNT) <> ".." THEN DCOUNT = DCOUNT + 1
    DIRARR(DCOUNT) = FINDNEXTFOLDER()
WEND

REM LOOP #2
REM FIND ALL *.VP FILES IN EACH DIRECTORY FOUND IN EARLIER LOOP
DIM I%
FCOUNT = 1
FOR I% = 1 TO DCOUNT-1
    FILESARR(FCOUNT) = FINDFIRSTFOLDER("D:\COREL\VENTURA\SAMPLES\" + DIRARR(I%) + "\*.VP", 1 OR
2 OR 4 OR 32 OR 128)
    WHILE (FILESARR(FCOUNT) <> "" )
    MESSAGE DIRARR(I%) & CHR(13) & FILESARR(FCOUNT)
         FCOUNT = FCOUNT + 1
         FILESARR(FCOUNT) = FINDNEXTFOLDER()
    WEND
```

```
NEXT I%
```

In the above example, the first loop fills an array (**DIRARR**) with the names of the normal folders in the D:\COREL\VENTURA\SAMPLES folder. The second loop searches the folders in the SAMPLES folder for any file with the extension VP. Any found VP file has its name added to the **FILESARR** array and has its name displayed in a message box.

The following statement in the first loop is used to remove the current (.) folder and parent (..) folder from being sent to the **DIRARR** array:

```
IF DirArr(Dcount) <> "." AND DirArr(Dcount) <> ".." THEN Dcount = Dcount + 1
```

{button ,AL(`CURRFOLDER;filemode;getfileattr;getcurrfolder;setcurrfolder;',0,"Defaultoverview",)} Related Topics

FREEFILE function

FREEFILE ( )

Returns the lowest file number not associated with an open text file in the computer's memory.

Example

```
OPEN "temp.out" FOR OUTPUT AS 1
OPEN "temp2.out" FOR OUTPUT AS 5
i% = FREEFILE( )
```

The lowest available file number in this example is 2 because 1 is already being used, therefore **i** is set to 2.

{button ,AL(`OPEN_APPEND;OPEN_INPUT;OPEN_OUTPUT;;;',0,"Defaultoverview",)} Related Topics

GETAPPHANDLE function

ReturnValue& = GETAPPHANDLE ( )

Returns the Application Instance Handle for the Corel application that is running a script. For example, if you are running the script from the Corel SCRIPT Editor, GETAPPHANDLE returns the Editor's Application Instance Handle. If you run a script from Corel VENTURA, GETAPPHANDLE returns VENTURA's Application Instance Handle. This function is used in conjunction with DLL calls that require the application's handle.

Return Value

Lets you specify a numeric variable that is passed the Application Instance Handle.

Example

```
hand = GETAPPHANDLE()
```

{button ,AL(`declare_lib;open_output;open_append;open_input;getwinhandle;getapphandle;',0,"Defaultoverview",)}
Related Topics

GETCOMMANDLINE function

ReturnValue$ = GETCOMMANDLINE ( )

Returns the parameters used in the command line that launch a script. To test this command, specify a command line in the Command Line text box in the Corel SCRIPT Editor's Options dialog box (click Tools, Options, Environment tab). For more information about command lines, click .

Return Value

String variable that is passed the command line parameters.

Example

```
CommandLine$ = GETCOMMANDLINE ( )
```

{button ,AL(`setcurrfolder;getCURRFOLDER;getfolder;ht_start_cse_custom;;',0,"Defaultoverview",)} Related Topics

GETCURRFOLDER function

ReturnValue$ = GETCURRFOLDER ( )

Returns the name of the active Windows folder and path.

Return Value

String variable that is passed the name of the active Windows folder and path.

Note

- You can set the active folder using the **SETCURRFOLDER** statement.
- In Corel SCRIPT version 7.0, the **GETCURRFOLDER** function and the **SETCURRFOLDER** statement replace the **CURRFOLDER** statement.

Example

```
SETCURRFOLDER "c:\corel\graphics8\scripts\"
folder$ = GETCURRFOLDER ( )
MyFile$ = "\MyScript.csc"
MyPathFile = folder$ & MyFile&
```

In the above example the first line sets the active folder. The second line assigns the active folder to a string variable. The third line assigns a file name to a string variable. In the last line a string variable is assigned a value which is made by combining the folder and file string variables.

{button ,AL(`setcurrfolder;getCURRFOLDER;getfolder;;;',0,"Defaultoverview",)} Related Topics

GETPROCESSINFO function

ReturnValue& **= GETPROCESSINFO (ProcessHandle)**

This function returns the status of an executable.

Return Value

Numeric variable that is passed a value that indicates whether an executable is running. If the executable is running, this variable is passed the value 259.

Parameter	Description
ProcessHandle	Numeric expression specifying the Windows Process Handle of an executable. Use the **STARTPROCESS** function to determine an executable's Windows Process Handle.

Example

```
launch = STARTPROCESS ("C:\WINDOWS\CALC.EXE")
... 'other script statements
... 'other script statements
... 'other script statements
Calc_Status = GETPROCESSINFO (launch)
```

The above example launches the Windows Calculator and passes the Windows Process Handle to the **launch** variable. The **launch** variable is used with the **GETPROCESSINFO** function to determine whether the Calculator is running.

{button ,AL(`STARTPROCESS ;GETPROCESSINFO;INCLUDE;INPUT;cs_exe_dll;;;',0,"Defaultoverview",)} Related Topics

GETSCRIPTFOLDER function

ReturnValue$ = GETSCRIPTFOLDER ( )

Returns the path and the folder where the executing script resides. If this function is used in a Corel SCRIPT Executable, it returns the path and folder where the Executable resides.

Return Value

String variable that is passed the path and folder of an executing script or Executable.

Note

- If the script has not been previously saved to disk or a network, this function returns an empty string.

Example

```
folder = GETSCRIPTFOLDER ( )
```

{button ,AL(`DECLARE_LIB;OPEN_OUTPUT;OPEN_INPUT;GETWINHANDLE;GETAPPHANDLE;Creating Corel SCRIPT Executables;;;',0,"Defaultoverview",)} Related Topics

GETTEMPFOLDER function

ReturnValue$ = GETTEMPFOLDER ( )

Returns the path and the folder of the system's Windows temporary folder.

Return Value

String variable that is passed the path and folder of the system's Windows temporary folder.

Example

```
t_folder = GETTEMPFOLDER ( )
```

{button ,AL(`GETCURRFOLDER;GETSCRIPTFOLDER;;;;',0,"Defaultoverview",)} **Related Topics**

GETVERSION function

ReturnValue& **= GETVERSION (option)**

Returns the system or Corel SCRIPT version numbers.

Return Value

The GETVERSION function returns one of the following values:

- **Option 0** - Corel SCRIPT run-time interpreter:

  Four-digit number. The first two digits represent the major version number and the last two digits represent the minor version number. For example, the four digit number 7001 indicates major version 7.0 and minor version 01.

- **Option 10** - Corel SCRIPT compiler version:

  Four-digit number. The first two digits represent the major version number and the last two digits represent the minor version number. For example, the four digit number 7001 indicates major version 7.0 and minor version 01.

- **Option 30** - Windows platform:

  Single-digit number indicating the Windows platform that Corel SCRIPT is being used with.

  0 Win32s (Windows 3.11)

  1 Windows 95

  2 Windows NT

- **Option 31** - Windows major version number:

  For example, if you're running Windows 95 and your Windows version number is 4.00.950, the major version number is 4. The same applies to Windows NT.

- **Option 32** - Windows minor version number:

  For example, if you're running Windows 95 and your Windows version number is 4.00.950, the minor version number is 0.

- **Option 33** - Windows build number:

  For example, if you're running Windows 95 and your Windows version number is 4.00.950, the minor version number is 950.

Parameter	Description
option	Lets you specify the system or Corel SCRIPT component to query: 0 Corel SCRIPT run-time interpreter version (the SCINT**xx**.DLL file being used with the current session of Corel SCRIPT) 10 Corel SCRIPT compiler version number 30 Windows platform 31 Windows major version number 32 Windows minor version number 33 Windows build number

Note

- For a script, Executable, DLL, or Corel Add-on created with Corel SCRIPT to run, the major version numbers for Corel SCRIPT (the compiler) and the Corel SCRIPT run-time interpreter (SCINT**xx**.DLL) must be the same or else an error will occur. A difference in the minor version numbers will not cause an error.

Example

```
CS_version = GETVERSION(10)
MESSAGE "Corel SCRIPT major version number " & LEFT (CS_version, 2)
MESSAGE "Corel SCRIPT minor version number " & RIGHT (CS_version, 2)
```

In the above example, the first line passes the Corel SCRIPT version number to **CS_version**. A message box is then used to display the first two digits in CS_version using the **LEFT** function. Next, a message box is used to display the first two digits in CS_version using the **RIGHT** function.

{button ,AL(`Creating Corel SCRIPT Executables;getscriptfolder;getapphandle;getwinhandle;;',0,"Defaultoverview",)}
Related Topics

GETWINHANDLE function

ReturnValue& = GETWINHANDLE ( )

Returns the window handle for the window that is running the script. For example, if you are running the script from the Corel SCRIPT Editor, **GETWINHANDLE** returns the Editor's Windows handle. If you run a script from CorelDRAW, **GETWINHANDLE** returns DRAW's Windows handle. This function is used in conjunction with DLL calls that require the window's handle.

Return Value

Lets you specify a numeric variable that is passed the window's handle.

Note

- If you run a Corel SCRIPT Executable, the **GETWINHANDLE** function returns 0. For more information about Corel SCRIPT Executables, click .

Example

```
hand = GETWINHANDLE()
```

{button ,AL(`declare_lib;open_output;open_append;open_input;getwinhandle;getapphandle;',0,"Defaultoverview",)}
Related Topics

INCLUDE statement

#INCLUDE filename

Lets you specify a Corel SCRIPT script to execute from the executing script. The specified script is treated as having its contents inserted into the executing script at the line holding the **INCLUDE** statement.

If you re-use many of the same constant, variable, and procedure declarations, you should consider putting that information in a separate script. Keeping this information in a separate script allows you to type the information once, and then call it as many times as you need with an INCLUDE statement in any new script you create.

For example, if you use the **WITHOBJECT** statement to call CorelDRAW 7or Corel VENTURA 7, you can insert the following statements in a INCLUDE file:

```
GLOBAL CONST DRAW7 = "CorelDraw.Automation.7"
GLOBAL CONST VENTURA7 = "CorelVentura.Automation.7"
```

The two statements above create two global constants named **DRAW6** and **VENTURA7**. If you always convert from tenths of a micron to another unit of measurement, you could include the following statement:

```
M_POINT = LENGTHCONVERT (1 , 3 , 1)
```

The **LENGTHCONVERT** statement creates a variable (**M_POINT**) that is equal to the number of tenths of a micron in a point.

Parameter	Description
filename	String expression specifying the filename, and optionally the path. If the path is not specified in **filename**, Corel SCRIPT will search for the file on your system in the following manner: 1 The active folder. You can use the **GETCURRFOLDER** statement to set or determine the active folder. 2 Folders in the path. The path is specified in the systems AUTOEXEC.BAT file. 3 Folders set in the Corel SCRIPT Editor's INCLUDE option. Click ▶ for more information about setting INCLUDE folders. 4 Folders specified with the **ADDFOL** statement.

Note

- Corel SCRIPT 7 introduces and includes CSI files. CSI are scripts (text files) that define commonly used constants. These files can be edited in the Corel SCRIPT Editor, and each Corel application that supports Corel SCRIPT has its own CSI file.

  Based on a typical Corel installation, an application's CSI file resides in the **C:\COREL*CorelSuite**application*\SCRIPTS** folder, where ***CorelSuite*** refers to the Corel products installed and ***application*** refers to the Corel application's folder. For example, the CorelDRAW 7 CSI file may reside in **C:\COREL\DRAW70\DRAW\SCRIPTS** folder and Corel VENTURA 7 CSI file may reside in the **C:\COREL\VENTURA7\VENTURA\SCRIPTS** folder.

  A general constants CSI file (SCPCONST.CSI) for Corel SCRIPT normally resides in the **C:\COREL*CorelSuite*\SCRIPTS** folder.

- The pound sign (#) is required in the syntax.

Example

```
#INCLUDE "My_constants.CSC"
#INCLUDE "SCPCONST.CSI"
```

The above example includes the Corel SCRIPT script MY_CONSTANTS.CSC and the SCPCONST constants file in the executing script.

{button ,AL(`GETCURRFOLDER;lengthconvert;angleconvert;addfol;const;;',0,"Defaultoverview",)} Related Topics

KILL statement

KILL fileName

Deletes a file. This statement is the same as clicking File, Delete in the Windows Explorer or in My Computer in Windows 95.

Parameter	Description
fileName	String expression specifying the filename to delete. You can use wild cards (* and ?) if you want to delete a group of files. For example, **script*.*** deletes all the files in the current folder beginning with **script**. Using **script?.*** deletes all the files in the current folder that begin with **script** and are followed by only one more character.

Note

- An open file cannot be deleted.

Example

```
KILL "temp.out"
```

Deletes the file TEMP.OUT in the current folder.

```
KILL "C:\MyDocs\temp.out"
```

Deletes the file TEMP.OUT in the **C:\MyDocs** folder.

{button ,AL(`rmfolder;open_output;;;;',0,"Defaultoverview",)} Related Topics

MKFOLDER statement and function

Statement: MKFOLDER folderName

Function: ReturnValue& = MKFOLDER (folderName)

Creates a new folder.

Return Value

The MKFOLDER function returns one of the following values:

- TRUE (-1) the folder was created
- FALSE (0) the folder was not created

Parameter	Description
folderName	String expression specifying the name of the folder to be created. Path information is optional.

Example

```
MKFOLDER "work"
```

Creates the folder **work** as a subfolder of the current folder.

```
success = MKFOLDER ("work")
```

Creates the folder **work** as a subfolder of the current folder and assigns -1 to **success**.

{button ,AL(`;GETCURRFOLDER;SETCURRFOLDER;RMFOLDER;;;;',0,"Defaultoverview",)} Related Topics

REGISTRYQUERY function

ReturnValue = REGISTRYQUERY (MainKey, SubKey, Value)

Returns the value data of a specified value key in the system's Windows registry. This function can help you determine where programs and files are installed on a user's system. This type of information is important when creating scripts that are to run on different system setups.

Return Value

Lets you specify the variable that is passed the value data of a specified value key in the Windows registry. Since this function can pass a string or numeric value, the variable you specify should be a variant. You can use the **GETTYPE** function to determine a variant's subtype.

Parameter	Description
MainKey	Lets you specify the main registry value key to query: 0 HKEY_CLASSES_ROOT 1 HKEY_CURRENT_USER 2 HKEY_LOCAL_MACHINE 3 HKEY_USERS 4 HKEY_PERFORMANCE_DATA 5 HKEY_CURRENT_CONFIG 6 HKEY_DYN_DATA
SubKey	Lets you specify the sub registry value key to query. This must be a complete key path. In Windows 95 for example, "SOFTWARE\Microsoft\Windows\CurrentVersion\Fonts" is a complete path.
Value	Lets you specify the registry value key to query. To specify a default value, use an empty string. Specify an empty string by using two quotation marks ("").

Note

- You cannot use this command to query binary values except those that can be converted to a number.

Example

```
Config_Ventura = REGISTRYQUERY (2, "SOFTWARE\Corel\Corel Ventura\7.0", "ConfigDir")
```

The above example returns the root folder where Corel VENTURA 7 is installed.

```
Arial_file = REGISTRYQUERY (2, "SOFTWARE\Microsoft\Windows\CurrentVersion\Fonts", "Arial
(TrueType)")
```

The above example returns the Arial True Type font's file name.

```
YourName$ = REGISTRYQUERY(2,"SOFTWARE\Corel","UserName")
```

The above example returns the name of the registered owner of Corel Software.

```
CompanyName$ = REGISTRYQUERY(2,"SOFTWARE\Corel","ORGANIZATION")
```

The above example returns the organization name of the registered owner of Corel Software.

```
Phone$ = REGISTRYQUERY(2,"SOFTWARE\Corel","PHONENUMBER")
```

The above example returns the phone number of the registered owner of Corel Software

{button ,AL(`GETSCRIPTFOLDER;GETAPPHANDLE;GETWINHANDLE;GETTYPE;;',0,"Defaultoverview",)} Related Topics

RENAME statement and function

RENAME file_folder1, file_folder2, overwrite

The RENAME statement changes the name of a file or folder, or can be used to move a file. You cannot move a folder using the RENAME statement.

You can also use RENAME as a function: it returns TRUE (-1) if the RENAME operation is successful, FALSE (0) if is not.

Parameter	Description
file_folder1	String expression specifying the name of the file or folder to move. **file_folder1** can include drive and folder path specifics ▸ if path specifics are not included, RENAME assumes the current folder.
file_folder2	String expression specifying the name of the file where **file_folder1** is to be moved. **file_folder2** can include drive and folder path specifics ▸ if path specifics are not included, RENAME assumes the current folder.
overwrite	If **file_folder2** already exists, determines whether to overwrite the existing file (you cannot overwrite existing folders): 0 = rename and overwrite 1 = overwrite fails (default if omitted)

Example

```
' statement example
DIM x AS STRING
DIM y AS STRING
x = "C:\work\example1.vp"
y = "D:\work\example1.vp"
RENAME x, y, 0
```

The above example moves the EXAMPLE1.VP file to the Work folder on the D drive.

```
' statement example
DIM x AS STRING
DIM y AS STRING
x = "C:\work\example1.cdr"
y = "C:\work\example2.cdr"
success = RENAME (x, y, 0)
```

The above example renames the EXAMPLE1.CDR file to EXAMPLE2.CDR, and assigns -1 to **success**.

{button ,AL(`FINDFIRSTFOLDER_FINDNEXTFOLDER;rename;copy;GETCURRFOLDER;rmfolder;SETcurrfolder;;',0,"Defaultoverview",)} Related Topics

RMFOLDER statement and function

RMFOLDER folderName

Removes an existing folder. The folder must be empty before it can be deleted. You can also use **RMFOLDER** as a function: it returns TRUE (-1) if the folder was removed, FALSE (0) if it was not.

Parameter	Description
folderName	String expression specifying the name of the folder to remove. **folderName** can include drive specifics ▶ if drive specifics are not included, RENAME assumes the current drive.

Example

```
RMFOLDER "C:\TEMP\WORK"
```

Removes the Work folder from the Temp folder.

```
success% = RMFOLDER "C:\TEMP\WORK"
```

Removes the Work folder from the Temp folder and assigns -1 to **success**.

{button ,AL(`KILL;MKFOLDER;GETCURRFOLDER;;;',0,"Defaultoverview",)} Related Topics

SETCURRFOLDER statement

SETCURRFOLDER FolderName

Sets the active Windows folder and path.

Parameter	Description
FolderName	String expression specifying a system folder and path.

Note

- In Corel SCRIPT version 7.0, the **SETCURRFOLDER** statement and the **GETCURRFOLDER** function replace the **CURRFOLDER** statement.

Example

```
SETCURRFOLDER "C:\corel\MyDocs\"
```

The above example sets the active folder and path to C:\COREL\MYDOCS\.

{button ,AL(`GETCURRFOLDER;getfolder;;;',0,"Defaultoverview",)} Related Topics

STARTPROCESS statement and function

Statement: STARTPROCESS exe

Function: ReturnValue& = STARTPROCESS (exe)

This statement or function launches executable files (.EXE files). You can also use this statement to launch Corel applications and Corel SCRIPT Executables.

Return Value

Numeric variable that is assigned a value indicating whether **STARTPROCESS** was not able to launch the specified executable. If the specified executable was not launched, 0 is assigned; otherwise the Windows Process Handle is returned.

Parameter	Description
exe	String expression specifying the executable to launch. The string expression should also include the executable's path and file extension.

Note

- Use the **GETPROCESSINFO** function to determine whether an executable is still running.

Example

```
STARTPROCESS "C:\WINDOWS\CALC.EXE"
```

The above example attempts to launch the Windows Calculator.

```
launch = STARTPROCESS ("C:\WINDOWS\CALC.EXE")
```

The above example attempts to launch the Windows Calculator and assigns a value to the **launch** variable, indicating whether the calculator was launched.

{button ,AL(`STARTPROCESS ;GETPROCESSINFO;INCLUDE;INPUT;cs_exe_dll;;;',0,"Defaultoverview",)} Related Topics

File Number

An integer (whole number) value between 1-10, inclusive. Non-integers are truncated.

CLOSE statement

CLOSE #num,...

Closes a text file opened with an OPEN statement (**OPEN...APPEND** or **OPEN...OUTPUT**).

Parameter	Description
num	Numeric expression(s) specifying a file number to close. If not specified, all open files are closed. The # sign is optional.

Note

- In your scripts, every OPEN statement should have a corresponding CLOSE statement.

Examples

```
CLOSE
```
Closes all open files.

```
CLOSE #1
```
Closes the file opened as 1.

```
CLOSE 1
```
Closes the file opened as 1.

```
CLOSE 1, 3, 5
```
Closes the files opened as 1, 3, and 5.

{button ,AL(`open_append;open_input;open_output;;;',0,"Defaultoverview",)} Related Topics

EOF function

EOF (#num)

Returns TRUE (-1) if the file pointer is at the end of an open text file in your computer's memory. Returns FALSE (0) if the file contains data beyond the pointer. The statement is often used to determine whether to continue processing a file.

Parameter	Description
#**num**	Numeric expression specifying an open file number to examine. The # sign is optional.

Example

```
IF EOF(1) THEN CLOSE 1
```

If the pointer is at the end of the file, then close the file.

{button ,AL(`lof;seek;open_append;print;write;',0,"Defaultoverview",)} Related Topics

INPUT function

INPUT(bytes, #num)

Starting at the pointer, reads a number of bytes (characters) from a text file.

Parameter	Description
bytes	Numeric expression specifying the number of bytes to be read.
#**num**	Numeric expression specifying an open file number to examine. The # sign is optional.

Example

```
myExtract$ = INPUT(50, #1)
```

Reads 50 characters from file 1 and assigns them to the variable **myExtract$**.

{button ,AL(`INPUT;LINE_INPUT;SEEK;FILEPOS;EOF;LOF;',0,"Defaultoverview",`main')} Related Topics

INPUT # statement

INPUT #num, var1, var2, ...

Reads from a file to a list of variables. Values in the file are separated by commas. Character strings that include commas must be enclosed in quotation marks. The quotation marks do not appear in the variable.

Parameter	Description
#**num**	Numeric expression specifying an open file number to examine.
var1, var2, ...	The variables that receive the fields as they are read from the file.

Notes

- The number sign (#) is required in the syntax.
- Numeric data with decimals can be input only if the period (**.**) is used as the decimal separator.
- If you're converting dates, they must be in the standard International format (yy/MM/dd hh:mm:ss).
- Booleans can be input only as TRUE or FALSE.

Example

```
INPUT #1, title$, number%
```

Reads a string and a numeric variable from file 1. Assigns the values to the string variable **title$** and the integer variable **number%**. The contents of file 1 could be in either of two formats:

```
"Ottawa", 50
"Huckleberry Finn", 101
"Hamlet", 220
```

or

```
"Ottawa", 50, "Huckleberry Finn", 101, "Hamlet", 220
```

{button ,AL(`input_dollar;open_input;line_input;write;;',0,"Defaultoverview",)} **Related Topics**

LINE INPUT statement

LINE INPUT #num, string

Reads the next line from an open text file in your computer's memory into a string.

Parameter	Description
#num	Numeric expression specifying an open file number to examine.
string$	String expression specifying the string variable to hold the line from the file.

Note

- The number sign (#) is required in the syntax.

Example

```
LINE INPUT #1, MyString$
```

Reads the next line from file 1 and places the string in the variable **MyString$**.

{button ,AL(`input_dollar;input;open_input;line_input;;;',0,"Defaultoverview",)} Related Topics

LOF function

LOF(#num)

LOF (Length of File) returns the number of bytes in an open text file.

Parameter	Description
#**num**	Numeric expression specifying an open file number to examine. The # sign is optional.

Example

```
bytes% = LOF(1)
```

Sets **bytes** to the number of bytes in file 1.

{button ,AL(`eof;seek;open_append;print;write;',0,"Defaultoverview",)} **Related Topics**

OPEN...APPEND statement

OPEN fileName FOR APPEND AS #num

Opens a text file to add or append sequential data to the end of the file. Data is appended using the **PRINT** or **WRITE** statement. The FOR clause is required.

Parameter	Description
fileName	String expression specifying a filename to open. If the file doesn't exist, it is created and then opened. Drive and folder information is optional.
#**num**	Numeric expression specifying a file number to associate with the open file. The # sign is optional.

Note

- You cannot use the **COPY** or **RENAME** command on an open file.

Example

```
OPEN "oldfile" FOR APPEND AS 3
```

Opens the file OLDFILE as file number 3 to append sequential data.

{button ,AL(`input_dollar;open_append;open_input;open_output;ACCESS;CLOSE;EOF;FILEMODE;FILEPOS;freefile;input;line_input;lof;print;seek;write',0,"Defaultoverview",`main')} Related Topics

OPEN...INPUT statement

OPEN fileName FOR INPUT AS #num

Opens a text file so that data can be read from it using the **INPUT** or **LINE INPUT** statement.

Parameter	Description
fileName	String expression specifying a filename to open. An error occurs if the file doesn't exist. Drive and folder information is optional.
#**num**	Numeric expression specifying a file number to associate with the open file. The # sign is optional.

Note

- You cannot use the **COPY** or **RENAME** command on an open file.

Example

```
OPEN FileString$ FOR INPUT AS 2
```

Opens the file specified by **FileString$** for input.

{button ,AL(`input_dollar;open_append;open_input;open_output;ACCESS;CLOSE;EOF;FILEMODE;FILEPOS;freefile;input;line_input;lof;print;seek;write',0,"Defaultoverview",`main')} Related Topics

OPEN...OUTPUT statement

OPEN fileName$ FOR OUTPUT AS #num%

Opens a text file so that data can be written into it using the **PRINT** or **WRITE** statement. The FOR clause is required.

Parameter	Description
fileName	String expression specifying a filename to open. If the file doesn't exist, it is created and then opened. Drive and folder information is optional.
#**num**	Numeric expression specifying a file number to associate with the open file. The # sign is optional.

Note

- You cannot use the **COPY** or **RENAME** command on an open file.

Example

```
OPEN "c:\temp\workfile.tmp" FOR OUTPUT AS 1
```

Opens the file C:\Temp\WORKFILE.TMP for output.

{button ,AL(`input_dollar;open_append;open_input;open_output;ACCESS;CLOSE;EOF;FILEMODE;FILEPOS;freefile;input;line_input;lof;print;seek;write',0,"Defaultoverview",`main')} Related Topics

PRINT statement

PRINT #num, expression1 { , | ; } expression2 { , | ; } ...

Prints expression(s) to an open text file (**OPEN...APPEND** or **OPEN...OUTPUT**) in your computer's memory.

Parameter	Description
#num	Numeric expression specifying a file number to print to.
expression1, expression2,...	The numeric or string expressions to print to the specified file.
{ , | ; }	Lets you specify how to separate the expressions being printed to the specified file. The comma places a tab between expressions. The semi-colon does not use a space, or a comma, between printed expressions.

Notes

- Ending a **PRINT** statement without a comma or a semicolon inserts a line return at the end of the printed expression.
- Strings are not enclosed in quotation marks when printed.
- Numeric expressions have either a leading space or a negative sign.
- The number sign (#) is required in the syntax.
- Dates and time are printed using the system's regional settings. The decimal character is also dependent on the system's regional settings. In Windows 95, click Start, Setting, Control Panel, Regional settings to view and change your system settings.
- Booleans are printed as TRUE or FALSE.

Example

```
PRINT #1, "A",          'inserts a tab after A
PRINT #1, "B"           'inserts a return after B
PRINT #1, "C";          'no spacing after C
PRINT #1, "D"
```

The above example prints the following to file #1 (there is a tab between A and B):

```
A   B
CD
```

In the following example, a blank line is printed, and then the string **HEADING:** is followed by the value of **string1$**, a comma, a space, and the value in **my_int%**.

```
PRINT #1,
PRINT #1, "HEADING: "; string1$; ", "; my_int%
```

{button ,AL(`seek;print;write;open_append;open_output;input;;',0,"Defaultoverview",)} Related Topics

SEEK statement

SEEK #num, position

To prepare for subsequent input, moves the file pointer to a specific byte position in an open text file in your computer's memory. Characters, spaces, tabs, and line breaks, are counted as characters.

Parameter	Description
num	Numeric expression specifying a file number to examine. The # sign is optional.
position	Numeric expression specifying the byte position to SEEK (1 is 1st byte).

Note

- The seek always starts from the first position in the file.
- The seek range is limited from 1 to length of the file (**LOF**) + 1.

Example

```
SEEK 1, 100
```

Moves the file pointer to the 100th byte in file 1.

{button ,AL(`OPEN_APPEND;OPEN_INPUT;OPEN_OUTPUT;lof;;',0,"Defaultoverview",)} Related Topics

WRITE statement

WRITE #num, expression1 { , | ; } expression2 { , | ; } ...

Writes expressions to an open text file (**OPEN...APPEND** or **OPEN...OUTPUT**) in your computer's memory.

Parameter	Description
#num%	Numeric expression specifying a file number to write to.
expression1, expression2,...	The numeric or string expressions to print to the specified file.
{ , | ; }	Lets you specify a separator between expressions. Both the comma and the semicolon insert a comma as a separator.

Notes

- Ending a complete **WRITE** statement without a comma or a semicolon inserts a line return at the end of the written expression.
- Strings are enclosed in quotation marks when printed.
- Numeric expressions have either a leading space or a negative sign.
- The number sign (#) is required in the syntax.
- Numeric data is written using the period (**.**) as a decimal separator.
- If you're converting dates, they must be in the standard International format (yy/MM/dd hh:mm:ss).
- Booleans are written as TRUE or FALSE.

Example

```
WRITE #1, "A ",          'inserts a space and comma after A
WRITE #1, "B"            'inserts a return after B
WRITE #1, "C";           'inserts a comma after C
WRITE #1, "D"            'inserts a return after D
WRITE #1,                'prints a blank line
```

The above example prints the following to file #1:

```
"A ","B"
"C","D"
```

{button ,AL(`seek;print;write;open_append;open_output;input;;',0,"Defaultoverview",)} Related Topics

ASC function

ASC(source)

Returns the numerical ANSI character value of the first character specified in a string. ASC is the opposite of the **CHR** function, which returns a character when the ANSI value is specified.

Parameter	Description
source	The string expression to be examined.

Note

- See the Corel SCRIPT Character Map for more ANSI details and Windows characters.

Example

```
i% = ASC("string")
```

This expression will assign the value 115, which is the ANSI value of the letter "s."

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

CHR function

CHR (value)

Returns the ANSI character that occupies the specified position in the ANSI character set. **CHR** is the opposite of **ASC**, which returns the ANSI code value when the character is entered. See the Corel SCRIPT Character Map for more ANSI details and Windows characters.

Parameter	Description
value	ANSI code value to be examined.

Example

```
s$ = CHR(65)
```

Assigns the letter "A" to the variable **s$**. (Character 65 of the ANSI character set is A.)

Special Characters

The CHR function is often used to add special characters to string variables that cannot be entered directly within double quotation marks. For example, to add double quotation marks to a string, you use character 34:

```
s$ = CHR(34) + "This will be in double quotes." + CHR(34)
MESSAGE s$
```

You can also use the function to add a return and a line feed within a string; use character 13 and 10, respectively:

```
s$ = "String 1" + CHR(13) + CHR(10) + "String 2"
MESSAGE s$
```

This will place the two strings on separate lines, as displayed in the message box.

The following table notes some of the special characters you can use with the **CHR** function.

Character Number	Special Character Returned
8	Backspace
9	Tab
10	Linefeed
13	Return
32	Space
34	Quotation mark

The following table notes some of the special characters you can use with the **CHR** function if you're using Corel VENTURA commands .**InsertSymbol** or .**TypeText**.

Character Number	Special Character Returned
10	Paragraph return
13	Forced line break
17	Em space
18	En space
19	Figure space
20	Thin space
21	Non-breaking space
22	Discretionary hyphen
34	Straight double quote
145	Typographical open single quote
146	Typographical close single quote
147	Typographical open double quote
148	Typographical close double quote

150	En dash
151	Em dash
153	Trademark
169	Copy right
174	Registered mark

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} **Related Topics**

INSTR function

INSTR (string1, string2, start**)**

Returns the starting position of the first occurrence of a string within another string. If the specified string is not found, the function returns 0.

Parameter	Description
string1	The string expression within which the search is made.
string2	The string for which you are searching.
start	Lets you specify the position where the search begins within **string1**. If unspecified, the search starts at the beginning of **string1** (same as **start**=1). Must be a non negative number and fractional numbers are rounded.

Example

```
pos = INSTR("Los Angeles", "Ang")
```

Sets **pos** to the value 5 because "Ang" occurs at the fifth character in the string "Los Angeles".

```
pos = INSTR("Los Angeles: City of Angels", "Ang", 8)
```

Sets **pos** to the value 22.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

LCASE function

LCASE(source)

Converts a string to lowercase characters.

Parameter	Description
source	The string expression to convert.

Note

- Non letter characters do not change when this function is used.
- You can use **UCASE** to convert to uppercase characters.

Example

```
x$="HI"
firststring$ = LCASE(x)
secondstring$ = LCASE("THerE")
MESSAGE firststring + " " + secondstring
```

The above example displays the converted strings "hi there" in a message box.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

LEFT function

LEFT (source, number)

Returns the specified number of characters from the beginning of a string.

Parameter	Description
source	The string expression from which the specified characters are returned.
number	Lets you specify the number of characters to be returned. Must be a non negative number and fractional numbers are rounded.

Note

- This function can be used to truncate user input from a dialog box.
- You can use the **LEN** function to determine the number of characters in a string.
- You can use the **RIGHT** function to return characters from the end of a string.

Example

```
abc$ = LEFT("I want to dance with you", 15)
MESSAGE abc$
```

Displays "I want to dance" in a message box.

Combine LEFT with INSTR to extract the portion of a string either up to or including a specified substring.

```
city$ = "San Francisco, California"
Mystr$ = LEFT(city$, INSTR(city$, ",")-1)
```

Extracts the characters in **city$** up to, but not including the comma, and places the result in the variable **Mystr$**. The variable **Mystr$** now has the value "San Francisco".

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

LEN function

LEN(source)

Returns the length or number of characters in a string.

Parameter	Description
source	The string expression that is measured.

Example

```
num = LEN("This is a test")
```

Assigns the length of the string, 14, to the variable **num**.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

LTRIM function

LTRIM(source)

Removes any leading spaces from a string. You can use **LTRIM** to remove leading spaces from dialog box inputs.

Parameter	Description
source	The string expression from which leading spaces are removed.

Example

```
MyString$ = "   Test"
MyString$ = LTRIM(MyString$)
```

Assigns "Test" to the variable **MyString$**. All leading spaces that were previously in the variable are removed.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

MID function and statement

Function: MID(source, index, count**)**
Statement: MID(source, index , count**) = modify**

If used as a function, MID returns a specified number of characters, starting at a specified position in a string. If used as a statement, MID replaces a portion of a string with another string, beginning at a specified character.

Parameter	Description
source	Any string, string variable, string constant, or expression returning a string. Hold the string to be modified.
source	For a function, the string expression from which to return characters. For a statement, the string expression holding the original string to be modified.
index	Position of the first character to be returned (function) or modified (statement).
count	For a function, the number of characters to be returned. For a statement, the number of characters to be overwritten. If not specified, the rest of **source** is returned or overwritten.
modify	A string expression replacing a portion of **source**.

Note

- You can use the **LEN** function to determine the number of characters in a string.

Example

```
s$ = MID("I want to dance with you", 11, 5)
```

The function extracts five characters from the string, beginning with the eleventh character. The variable **s$** then becomes "dance".

```
str1$ = "I want to dance with you"
MID(str1$, 22, 3) = "him"
```

The statement changes three characters, starting with the 22nd character, to the new string. The **str1$** variable now contains "I want to dance with him".

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} **Related Topics**

RIGHT function

RIGHT (source, number)

Returns the specified number of characters from the end of a string.

Parameter	Description
source	The string expression from which the specified characters are returned.
number	Lets you specify the number of characters to be returned. Must be a non negative number and fractional numbers are rounded.

Note

- This function can be used to truncate user input from a dialog box.
- You can use the **LEN** function to determine the number of characters in a string.
- You can use the **LEFT** function to return characters from the beginning of a string.

Example

```
abc$ = RIGHT("I don't want to dance", 13)
MESSAGE abc$
```

Displays "want to dance" in a message dialog box.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

RTRIM function

RTRIM(source)

Removes any trailing spaces from a string. You can use RTRIM to remove trailing spaces from dialog box inputs.

Parameter	Description
source	The string expression from which trailing spaces are removed.

Example

```
MyString$ = "Test   "
MyString$ = RTRIM(MyString$)
```

Assigns "Test" to the variable **MyString$**. All trailing spaces that were previously in the variable are removed.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

SPACE function

SPACE (num)

Returns a string that consists of a specified number of spaces (ANSI character number 32).

Parameter	Description
num	Lets you specify the number of spaces to be included in a string.

Note

- See the Corel SCRIPT Character Map for more ANSI details and Windows characters.

Example

```
Mystr$ = SPACE(4) + "Test" + SPACE(4)
```

Makes the string variable **Mystr$** equal to " Test ". (The string **Mystr$** now consists of 4 spaces, the word TEST and another 4 spaces).

```
x1 = "Corel"
x2 = "SCRIPT"
x3 = x1 + SPACE(1) + x2
```

Makes the string variable **x3$** equal to "Corel SCRIPT".

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

STR function

STR(num)

Returns a string representation of a numeric data type. The **STR** function is useful when you want to manipulate a number as a string.

Parameter	Description
num	Lets you specify the numeric expression that is returned as string representation.

Note

- If a positive number is converted, the **STR** function inserts a leading space before the first character. If a negative number is converted, the **STR** function inserts a negative sign before the first character.
- You can use only the period as a decimal separator with the **STR** function. If you're not using a period (**.**) as a decimal separator, the **CSTR** function can be used to convert a number to a string. Your Windows decimal settings are set in the Control Panel.
- If you're converting dates, they must be in the standard International format (yy/MM/dd hh:mm:ss).

Example

```
aInteger$ = STR(72)
aNonInteger$ = STR(.140166)
```

The first example assigns "72" to the variable **aInteger$**. The second example assigns "0.140166" to **aNonInteger$**.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

UCASE function

UCASE(source)

Converts a string to uppercase characters.

Parameter	Description
source	The string expression to convert.

Note

- Non letter characters do not change when this function is used.
- You can use **LCASE** to convert to lowercase characters.

Example

```
x$="hI"
firststring$ = UCASE (x)
secondstring$ = UCASE("THerE")
MESSAGE firststring + " " + secondstring
```

Displays the converted strings "HI THERE" in a message dialog box.

{button ,AL(`cs_strings_fns;;;;;',0,"Defaultoverview",)} Related Topics

VAL function

VAL (chars)

Converts a string to a number. The number's variable type is double. This function is the opposite of the **STR** function.

Parameter	Description
chars	The string expression to be converted. If the string does not begin with a number, VAL returns 0.

Note

- You can use only the period as a decimal separator with the VAL function. If you're not using a period (**.**) as a decimal separator, the **CDBL** or **CSNG** function can be used to convert a string to a number. Your Windows decimal settings are set in the Control Panel.
- The **VAL** function converts the string up to the first non-number character it encounters, from left to right in the string. Spaces are ignored.
- Because text box controls in dialog boxes can only return strings (even if numbers are input), you can use the **VAL** function to convert strings entered in dialog boxes to numbers.

Example

```
g = VAL("72nd Street")
h = VAL("72.700113")
```

Both the above statements assign 72 to the variables **g** and **h**.

{button ,AL(`cs_strings_fns;inputbox;;;;',0,"Defaultoverview",)} Related Topics

CBOL function

CBOL(NumStrExp)

Converts an expression to a Boolean data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Note

- A numeric expression equal to 0 is converted to FALSE. All other numeric expressions are converted to TRUE (-1).

Example

```
x% = 354.43
y = CBOL(x)
```

This example sets **y** to TRUE (-1).

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;;;;',0,"Defaultoverview",)} Related Topics

CCUR function

CCUR(NumStrExp)

Converts an expression to a Currency data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- When Corel SCRIPT converts the value to the currency data type, it rounds off the value, rather than truncating it.
- If the result of this function lies outside the Currency data type range, an error occurs.

Example

```
x = 354.432675434
y = CCUR(x*2)
```

This example sets **y** to a currency data type.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;;;;',0,"Defaultoverview",)} Related Topics

CDAT function

CDAT(NumStrExp)

Converts an expression to a Date data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- This function denotes a base date of December 31, 1899 at 12:00:00 A.M. as 1. Each additional whole number is one additional day. Each additional fraction is a portion of a day.
- When Corel SCRIPT converts the value to a date, it rounds off the value, rather than truncating it.

Example

```
x% = 25.25
y = CDAT(x)
```

This example sets **y** to January 24th, 1900 at 6:00:00 A.M.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;;;;',0,"Defaultoverview",)} Related Topics

CDBL function

CDBL(NumStrExp)

Converts an expression to a Double data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- When Corel SCRIPT converts the value to a double, it rounds off the value, rather than truncating it.
- If the result of this function lies outside the Double data type range, an error occurs.

Example

```
x@ = 35489097326.43    'x is currency data type
y = CDBL(x/2)
```

This example sets **y** to double data type.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;;;;',0,"Defaultoverview",)} Related Topics

CINT function

CINT(NumStrExp)

Converts an expression to a Integer data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- In converting the value to an integer, Corel SCRIPT rounds off the value, rather than truncating it. If the decimal portion of the number is 0.5, CINT rounds to the nearest even number. For example, 8.5 rounds to 8, and 9.5 rounds to 10.
- If the result of this function lies outside the Integer data type range, an error occurs.
- You can also use INT and FIX to remove the fractional portion of a number.

Example

```
x = 354.63
y = CINT(x)
```

This example sets **y** to 355.

```
xx = 354.43
yy = CINT(xx)
```

This example sets **yy** to 354.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;int;fix;;',0,"Defaultoverview",)} Related Topics

CLNG function

CLNG(NumStrExp)

Converts an expression to a Long data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- In converting the value to a long integer, Corel SCRIPT rounds off the value, rather than truncating it.
- If the result of this function lies outside the Long data type range, an error occurs.
- You can also use INT and FIX to remove the fractional portion of a number.

Example

```
x = 98765578.43
y = CLNG(x)
```

This example sets **y** to 98765578.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;int;fix;;',0,"Defaultoverview",)} Related Topics

CONST statement

GLOBAL **CONST constant**{%|&|!|#|@|$} **= expression** , constant{%|&|!|#|@|$} = expression, ...
GLOBAL **CONST constant** AS type **= expression**, constant AS type = expression, ...

Declares constants for use in place of numeric or string values. Changing the value of a constant normally requires editing only one script statement.

Parameter	Description
GLOBAL	An optional parameter used to declare global constants. Global constants are available to all procedures in a script. If not used, the constant is only available to the procedure in which it was declared.
constant{%|&|!|#|@|$}	Lets you specify the name of the constant. It follows the Corel SCRIPT naming convention. Optionally, a type-declaration character can follow the name.
constant	Lets you specify the name of the constant. It follows the Corel SCRIPT naming convention.
AS type	Declares the data type of the constant with a type declaration name.
expression	A numeric or string expression assigned to the declared constant.

Notes

- You cannot declare a constant as a variant.
- Unlike variables, you can't change or assign a new value to a constant once it has been declared.
- Constants can be used to declare the size of an array.
- If you re-use many of the same constants, you should consider putting them in a separate script. Keeping this information in a separate script allows you to type the information once and then call it as many times as you need with an **INCLUDE** statement in any new script you create.

Example

```
REM creates a global constant for the base of the natural logarithm
GLOBAL CONST NATURAL_LOG# = 2.71828182845

REM creates a local constant for pi
CONST PI AS DOUBLE = 3.1415926535
```

{button ,AL(`script_procedures;global;dim;using_variables;Using_constants;corel_script_data_type_summary;;;',0,"Defaultoverview",)} Related Topics

CSNG function

CSNG(NumStrExp)

Converts an expression to a Single data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- In converting the value to a single, Corel SCRIPT rounds off the value, rather than truncating it.
- If the result of this function lies outside the Currency data type range, an error occurs.

Example

```
x = 354
y = CSNG(x + 0.02)
```

This example sets **y** to a single data type.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;;;;',0,"Defaultoverview",)} Related Topics

CSTR function

CSTR(NumStrExp)

Converts an expression to a String data type. Corel SCRIPT data type conversion functions set the result of an expression to a specified data type rather than the default data type.

Parameter	Description
NumStrExp	A numeric or string expression.

Notes

- A Boolean value of 0, when cast as a string, will return "FALSE". All other values will return "TRUE".
- Converting a date returns a string with a date in the Windows long-date format.
- You can also use **STR** to convert numeric expressions to a string.
- The **CSTR** function is locale aware; that is, it uses the decimal character specified in the user system's regional settings. In Windows 95, click Start, Setting, Control Panel, Regional settings to view and change your system settings.

Example

```
x = 354.43
y = CSTR(x)
```

This example sets **y** to "354.43", a string.

{button ,AL(`Corel_SCRIPT_data_type_summary;vars_convert;STR;;;',0,"Defaultoverview",)} Related Topics

DEFINE statement

#DEFINE substituteID syntax : syntax: ...

This statement is used to create a substitute for Corel SCRIPT syntax. Using a substitute is similar to using a constant. A substitute can be used to provide meaningful names for numeric and string values but it can also be used to replace script statements and functions.

Parameter	Description
substituteID	Lets you specify the identifier for the substituted Corel SCRIPT syntax.
syntax : syntax: ...	Lets you specify the Corel SCRIPT to substitute.

Notes

- You can remove a defined substitute from a script by using the **UNDEF** statement.
- The number sign (#) is required in the syntax.

Example

```
#DEFINE ErrorBox MESSAGE "An error occurred" : MESSAGE "FIX IT!"
```

The above example replaces two MESSAGE statements with the substitute **ErrorBox**.

{button ,AL(`const;global;define;undef;;',0,"Defaultoverview",)} Related Topics

DIM statement

Syntax for variables:
DIM variable{%|&|!|#|@|$}, variable AS type,...

Syntax for arrays:
DIM array_name{%|&|!|#|@|$} **(upperbound)**
DIM array_name(upperbound) AS type
DIM array_name{%|&|!|#|@|$} **(lowerbound TO upperbound)**
DIM array_name(lowerbound TO upperbound) AS type

Use **DIM** to declare local variables explicitly or to specify the number and type of elements in an array. Local variables are available only to one procedure in a script.

Parameter	Description
variable{%|&|!|#|@|$}	
or **variable AS type**	Lets you specify the name of the variable and follows the Corel SCRIPT naming convention. The variable can be declared using a type-declaration character (the first option)or by using a type declaration name (the second option).
array_name {%|&|!|#|@|$}	
or **array_name AS type**	Lets you specify the name of the array and follows the Corel SCRIPT naming convention. The array can be declared using a type-declaration character (the first option)or by using a type declaration name (the second option).
upperbound	The upper bound of the array expressed as an integer. If you do not use a TO clause to specify the number of array elements, the default (1 TO **upperbound**) is used.
lowerbound	The lower bound of the array expressed as an integer. If you do not use a TO clause to specify the number of array elements, the default (1 TO **upperbound**) is used.

Note

- If you declare a variable or array using the **DIM** statement without a type declaration name or a type-declaration character, Corel SCRIPT sets the variable or array to a variant data type.
- Variables and arrays declared in the main section of a script are available in the main section only. Global variables are available to all procedures in a script. See **GLOBAL** for more information.
- Declaring a variable in a subroutine or a function makes it available only in the procedure it was declared. See Variable availability for more information about using variables in procedures.
- It's a generally accepted programming convention to put declaration statements at the beginning of a procedure (main section, subroutines, or functions).
- The **DIM** statement can be placed anywhere in a script before the variable(s) it declares is called.
- Arrays can hold only one data type. The number of elements arrays can hold is limited to your system's memory.
- See Multi-dimensional arrays to create arrays of more than one dimension.
- You can re-dimension (or resize) arrays using the **REDIM** statement.

{button ,AL(`Explicitly_declaring;dim;redim;global;using_arrays;using_variables;lbound;ubound;multi_dimensional_arrays;',0,"Defaultoverview",)} Related Topics

GETTYPE function

GETTYPE(expression)

Returns an expression's data type. In the case of variants, the data subtype is returned. See Variants for more information.

Return Value

The GETTYPE function returns one of the following values:

- 0 Empty variant
- 2 Boolean
- 3 Integer
- 4 Long
- 5 Single
- 6 Double
- 7 Date
- 8 Currency
- 9 String

Parameter	Description
expression	Lets you specify the expression to use.

Example

```
x% = 66    'integer data type
y# = 14    'long data type
Z = x / y
A = GETTYPE (x / y)
```

In the above example, **A** is assigned the value 2 since the variant **Z** data subtype is set to long. In the following example, **B** is set to 4 since a whole number is treated as a long and **C** is set to 6 since a fractional number is treated as a double.

```
B = GETTYPE (3)
C = GETTYPE (3.3)
```

{button ,AL(`all_vars;dim;;;;',0,"Defaultoverview",)} **Related Topics**

GLOBAL statement

Syntax for variables
GLOBAL variable{%|&|!|#|@|$}, variable AS type,...

Syntax for arrays
GLOBAL array_name{%|&|!|#|@|$} **(upperbound)**
GLOBAL array_name(upperbound) AS type
GLOBAL array_name{%|&|!|#|@|$} **(lowerbound TO upperbound)**
GLOBAL array_name(lowerbound TO upperbound) AS type

Use GLOBAL to explicitly declare variables or to specify the number and type of elements in an array. Global variables and arrays are available to all procedures in a script. See the **DIM** statement for information on local variable declarations. See Variable availability for more information on the use of variables in procedures.

Parameter	Description
variable{%|&|!|#|@|$}	
or **variable AS type**	Lets you specify the name of the variable and follows the Corel SCRIPT naming convention. The variable can be declared using a type-declaration character (the first option)or by using a type declaration name (the second option).
array_name {%|&|!|#|@|$}	
or **array_name AS type**	Lets you specify the name of the array and follows the Corel SCRIPT naming convention. The array can be declared using a type-declaration character (the first option)or by using a type declaration name (the second option).
upperbound	The upper bound of the array expressed as an integer. If you do not use a TO clause to specify the number of array elements, the default (1 TO **upperbound**) is used.
lowerbound	The lower bound of the array expressed as an integer.

Notes

- If you declare a variable or array using the **GLOBAL** statement without a type declaration name or a type-declaration character, Corel SCRIPT sets the variable or array to a variant data type.
- Global variables cannot be declared in a subroutine or a function. Additionally, globals cannot be declared in a flow construct such as **FOR...NEXT** or **DO...LOOP**. This restriction also applies to arrays.
- It's a generally accepted programming convention to put declaration statements at the beginning of the script.
- Arrays can hold only one data type. The number of elements arrays can hold is limited by your system memory.
- You can also declare global constants. See **CONST** for more information.
- See Multi-dimensional arrays to create arrays of more than one dimension.

{button ,AL(`const;dim;global;using_arrays;using_variables;lbound;ubound;multi_dimensional_arrays;',0,"Defaultoverview",)} Related Topics

Examples for DIM statement

Variables

```
DIM my_color$
DIM my_color AS STRING
```

The above examples show different methods of declaring variables. The above DIM statements all declare strings.

```
DIM a AS INTEGER, b AS BOOLEAN, c AS SINGLE
```

You can also mix the type of variables you declare with a DIM statement.

Arrays

```
DIM color$(5)
color$(1) = "black"
color$(2) = "red"
color$(3) = "white"
color$(4) = "blue"
color$(5) = "green"
```

Creates a string array named **color$** that consists of 5 elements.

```
DIM salespeople(-2 TO +3) AS INTEGER
salespeople(-2) = 1
salespeople(-1) = 3
salespeople(0) = 5
salespeople(1) = 7
salespeople(2) = 9
salespeople(3) = 11
```

Creates an integer array named salespeople that consists of 6 elements.

Note

- See Multi-dimensional arrays to create arrays of more than one dimension.

{button ,AL(`example_multi_array;dim;global;using_variables;using_arrays;',0,"Defaultoverview",)} Related Topics

Examples for GLOBAL statement

Variables

```
GLOBAL my_color$
GLOBAL my_color AS STRING
```

The above examples show different methods of declaring global variables. The above GLOBAL statements all declare strings.

```
GLOBAL a AS INTEGER, b AS BOOLEAN, c AS SINGLE
```

You can also mix the type of variables you declare with a GLOBAL statement.

Arrays

```
GLOBAL color$(5)
color$(1) = "black"
color$(2) = "red"
color$(3) = "white"
color$(4) = "blue"
color$(5) = "green"
```

Creates a global string array named **color$** that consists of 5 elements.

```
GLOBAL salespeople(-2 TO +3) AS INTEGER
salespeople(-2) = 1
salespeople(-1) = 3
salespeople(0) = 5
salespeople(1) = 7
salespeople(2) = 9
salespeople(3) = 11
```

Creates an global integer array named salespeople that consists of 6 elements.

Note

- See Multi-dimensional arrays to create arrays of more than one dimension.

{button ,AL(`example_multi_array;dim;global;using_variables;using_arrays;',0,"Defaultoverview",)} Related Topics

LBOUND function

LBOUND(array, dimension**)**

Returns the lower bound for a specified dimension of an array.

Parameter	Description
array	Lets you specify the array to dimension.
dimension	A whole number variable or numeric constant ranging from 1 to the number of dimensions in the array. Lets you specify which dimension's lower bound is returned. If omitted, the limit of the first dimension is returned.

Example

```
DIM a%(-5 TO 7, 10)
x% = LBOUND(a%,1)
y% = LBOUND(a%,2)
```

Sets **x%** and **y%** to -5 and 1, respectively.

{button ,AL(`LBOUND;UBOUND;DIM;USING_ARRAYS;multi_dimensional_arrays;',0,"Defaultoverview",)} **Related Topics**

LET statement

LET **variable**{%|&|!|#|@|$} **= expression**
LET **variable = expression** AS type
variable{%|&|!|#|@|$} **= expression**
variable = expression AS type

Assigns the value of an expression to a variable. The **LET** keyword is optional.

Parameter	Description
variable{%|&|!|#|@|$}	Lets you specify the name of variable and is assigned **expression**'s value. The variable name follows the Corel SCRIPT naming convention.
variable	Lets you specify the name of variable and is assigned **expression**'s value. The variable name follows the Corel SCRIPT naming convention.
expression	A numeric or string expression that is assigned to the variable.
type	Declares the variable's type with a type declaration name.

Note

- There isn't an advantage in using the LET statement to assign an expression to a variable, but in some cases it can make your script easier to read and modify.
- If the variable's data type is not declared, the variable is set to the variant data type.

Example

```
LET stringVar$ = "This is a string."
```

Assigns the string "This is a string." to the variable **stringVar$**.

```
stringVar$ = "This is a string."
```

Assigns the string "This is a string." to the variable **stringVar$**. The LET keyword is omitted.

```
result% = (a% + b%) / c%
```

Assigns the result of the sum of the values of variables **a**% and **b%**, divided by the value of **c%**, to the variable **result%**. The LET keyword is omitted.

{button ,AL(`variable_availability;using_variables;Dim;;;',0,"Defaultoverview",)} Related Topics

REDIM statement

REDIM PRESERVE **array_name**{%|&|!|#|@|$} **(upperbound)**
REDIM PRESERVE **array_name(upperbound)** AS type
REDIM PRESERVE **array_name**{%|&|!|#|@|$} **(lowerbound TO upperbound)**
REDIM PRESERVE **array_name(lowerbound TO upperbound)** AS type

Used to re-dimension (change the number of array elements or dimensions) in a previously declared array. See the **DIM** statement for information about declaring arrays.

Parameter	Description
array_name {%|&|!|#|@|$}	
or **array_name AS type**	Lets you specify the name of the array and follows the Corel SCRIPT naming convention. The array can be declared using a type-declaration character (the first option)or by using a type declaration name (the second option).
upperbound	The upper bound of the array expressed as an integer. If you do not use a TO clause to specify the number of array elements, the default (1 TO **upperbound**) is used.
lowerbound	The lower bound of the array expressed as an integer. If you do not use a TO clause to specify the number of array elements, the default (1 TO **upperbound**) is used.
PRESERVE	Corel SCRIPT keyword which when used indicates to preserve the data in the array that is being re-dimensioned. Use the PRESERVE keyword to retain data in the elements that are part of the re-dimensioned array. If this keyword is not used, all the array data is discarded.

Note

- You can't use the **REDIM** statement to change the variable type an array can hold.

Example

```
DIM color$(5)
color$(1) = "black"
color$(2) = "red"
color$(3) = "white"
color$(4) = "blue"
color$(5) = "green"
```

The above example creates a string array with 5 elements. The following scripting command removes the last two elements from the **color** string array by using the REDIM statement.

```
REDIM PRESERVE color$(3)
```

{button ,AL(`dim;redim;global;using_arrays;using_variables;lbound;ubound;multi_dimensional_arrays;',0,"Defaultoverview",)} Related Topics

SETEMPTY statement

SETEMPTY variable

This statement removes data from a variable. If the variable is numeric, the variable is set to 0. If the variable is a string, the variable is set to an empty string (""). If the variable is a variant, the variable is set to Empty. An Empty variant contains no valid data in both a numeric and string context. See Corel SCRIPT data type summary for more information about data types and variants.

Parameter	Description
variable	Lets you specify the name of the variable to remove data from.

Example

```
SETEMPTY VariableName
```

The above example remove data from the **VariableName** variable.

{button ,AL(`variants;variable_availability;using_variables;Dim;;;',0,"Defaultoverview",)} Related Topics

STATIC statement

STATIC variable{%|&|!|#|@|$} = expression
STATIC variable AS type = expression

This statement declares and assigns an initial value to a variable in a user-defined subroutine or function. Static variables can only be called in these user-defined procedures, and retain their values as long as the script they are declared in is running. See Variable availability for more information about using variables in procedures.

Parameter	Description
variable{%|&|!|#|@|$}	Lets you specify the name of the variable to be declared. The variable name follows the Corel SCRIPT naming convention.
variable	Lets you specify the name of the variable to be declared. The variable name follows the Corel SCRIPT naming convention.
type	Declares the variable's data type with a type declaration name.
expression	The numeric expression initially assigned to the declared static variable. This option cannot be used with static variables of string or variant data types. Declaring and assigning a value to constant at the same time is new to Corel SCRIPT in version 7.0.

Note

- If you declare a variable using the **STATIC** statement without a type declaration name or a type-declaration character, Corel SCRIPT sets the variable to a variant data type.
- It's a generally accepted programming convention to put static declaration statements at the beginning of subroutines or functions with **DIM** declarations.

{button ,AL(`function_end_function;sub_end_sub;variable_availability;using_variables;Dim;;;',0,"Defaultoverview",)}
Related Topics

Example for STATIC statement

```
REM main section of script file
DECLARE FUNCTION staticFunc% (a%)
FOR i% = 1 to 5
    j% = staticFunc%(i%)
NEXT I%
'
REM (Static Function Example)
FUNCTION staticFunc%(a%)
STATIC staticVar%
' Because staticVar% is STATIC, it retains its previous value
' each time the function is called
staticVar% = staticVar% + a%
' The function returns the current value of staticVar%
staticFunc% = staticVar%
END SUB
```

The variable **staticVar%** in the function is created as a STATIC variable, so that its value remains unchanged each time the function is called. In the main program, a FOR loop calls the function five times. The result of each function call follows:

1 The first time the script runs, **staticVar%** has a value of 0 because it is created for the first time. The passed parameter, **i%**, has a value of 1, and the variable also has a value of 1.

2 In the second call, **staticVar%** has a value of 1 and the passed parameter has a value of 2. So the calculation causes **staticVar%** to be 3.

3 In the third call, **staticVar%** is equal to 3 and **i%** is equal to 3, so **staticVar%** has a new value of 6.

4 In the fourth call, **staticVar%** is 6 and **i%** is 4, giving **staticVar%** a new value of 10.

5 In the last call, **staticVar%** is 10 and **i%** is 5, giving **staticVar%** a value of 15.

{button ,AL(`example_vars;;;;;',0,"Defaultoverview",)} Related Topics

UBOUND function

UBOUND(array, dimension**)**

Returns the upper bound for a specified dimension of an array.

Parameter	Description
array	Lets you specify the array to dimension.
dimension	A whole number variable or numeric constant ranging from 1 to the number of dimensions in the array. Lets you specify which dimension's upper bound is returned. If omitted, the limit of the first dimension is returned.

Example

```
DIM a%(-5 TO 7, 10)
x% = UBOUND(a%,1)
y% = UBOUND(a%,2)
```

Sets **x%** and **y%** to 7 and 10, respectively.

{button ,AL(`redim;LBOUND;UBOUND;DIM;USING_ARRAYS;multi_dimensional_arrays;',0,"Defaultoverview",)} Related Topics

UNDEF statement

#UNDEF substituteID

This statement is used to remove a substitute declared with a **DEFINE** statement from a script.

Parameter	Description
substituteID	Lets you specify the identifier of the substituted Corel SCRIPT syntax to remove.

Note

- The number sign (#) is required in the syntax.

Example

```
#UNDEF ErrorBox
```

The above example removes the substitute **ErrorBox**.

{button ,AL(`const;global;define;undef;;',0,"Defaultoverview",)} **Related Topics**