What's new

Delphi 6 introduces new features and enhancements in the following areas:

- IDE
- Internet
- XML
- Compiler
- COM/Active X
- Database support
- CORBA
- Actions
- Custom Variants
- VCL units and features
- RTL units and features
- Cross-platform development
- Translation tools
- Deployment changes
- Help system extensibility

Note: Some features are not available in all editions of Delphi.

New IDE features

The IDE has new features in the following areas:

- Data modules
- Object TreeView
- Code editor
- Object Inspector
- Code Insight tools
- Project Manager
- File menu
- New Items dialog box
- Internet toolbar
- Component palette changes
- Key mapping modules
- Environment Options dialog box
- Directories dialog box
- Context menu display
- Writing design-time packages

Data modules (All editions)

The Data Module Designer in Delphi 5 has been divided into three parts:

1. A standard data module (as in Delphi 4), replacing the Components page.
2. The Object TreeView, replacing the Data Module Designer's left pane.
3. The Diagram page, replacing the Data Diagram page.

Object TreeView (All editions)

Located in the upper left-hand corner of the IDE, the TreeView is a tree diagram that displays the logical relationships between visual and nonvisual components on a form, data module, or frame.

The TreeView is synchronized with the Object Inspector and Form Designer so, if you select an item and change the focus in any one of these three tools, the focus changes in the other two tools. Furthermore, the Diagram page on the Code editor works only by dragging and dropping elements from the Object TreeView.

The Object TreeView has some added functionality:

- The TreeView is positioned above the Object Inspector and if not displayed, appears when you press Alt+Shift+F11 or choose View|Object TreeView.
- The TreeView includes components on a form and a frame as well as a data module.
- The TreeView includes visual as well as nonvisual components.
- A new toolbar includes New Item, Delete, and Move Up/Down buttons. These buttons work for component properties. For example, if you've added a dataset component, you can select the Aggregates property and click the New Item button to add a field. If there is more than one type of thing that can be added, the New Item button will drop down a menu to pick from.
- Additional glyphs represent the type of item, such as a visual parent or child component, nonvisual component, or implicit or unimplicit property. For example, items implicitly created for you behind the scenes, such as a default session, are black-and-white.
- Before, when a component's property values were not completed, its glyph was surrounded by a red circle. Now, a red question mark enclosed in a yellow circle appears to the left of the icon.
- Right-clicking will display more options than what appeared in the Data Module Designer.

Code editor

Surface designers (Professional and Enterprise editions)

The Code editor now supports surface designers, or package-loaded custom views. These views are accessible as tabs, or pages, located on the status bar. The only built-in view is the standard Code page. Depending on which edition of Delphi you have, you can access the following Code editor pages:

Diagram page (Professional and Enterprise editions)

The Diagram page on the Code editor provides visual tools for setting up a diagram of boxes and lines to display relationships among visual and nonvisual components. It is a documentation tool, since it illustrates these relationships schematically and lets you add comments to the diagram. Components do not appear on the Diagram page until you drag them from the Object TreeView.

New features:

- You can select multiple items from the Object TreeView and drag them to the Diagram page at one time.
- The left side of the Diagram page has an edit box where you can type a name and description for each diagram you create and a drop-down list box to find previously named diagrams.
- The toolbar of buttons for connector relationships and comment blocks sits at the top of the Diagram page.
- Property connectors are labeled automatically. You can drag the labels elsewhere on the page.
- You can create a diagram for each data module, form, or frame you've added to your project.

WebSnap pages (Enterprise edition)

When creating a Web server application using WebSnap, the components in the Web page module generate pages for HTML Script, HTML Result, Preview, XML Tree, and XSL Tree.

Drag-and-drop tabs (All editions)

The unit tabs on top of the Code editor and be reorganized by dragging and dropping them. For example, if you have two units called About and TextEditor, you can drag About to the right of TextEditor.

Object Inspector (All editions)

The Object Inspector is now located below the Object TreeView.

New features:

Instance list box

The instance drop-down list box is at the top of the Object Inspector:

New features:

- The instance list displays the class name for each object in the list, not just the one at the top.
- You can give a component the same name as the form or data module that it is on. For example, you can add a button component to Form1 and rename the button as Form1; both names appear in the instance list.
- The instance list box displays a tooltip for the selected component, which is useful if a component name is wider than the instance list.
- The instance list box can be hidden.

Properties dialog box

The Object Inspector's context menu has a new Properties dialog box, which can also be accessed by choosing Tools|Environment Options and clicking the Object Inspector page. It includes several display options:

- SpeedSettings to customize the colors of the Object Inspector.
- Displaying or hiding the instance list, class names in the instance list, status bar, background grid, and read-only properties.
- Properties of component references can be expanded inline and displayed on both the properties and events pages.

Expanded inline component references

Expanded inline component references display the properties and events of a referenced component

without having to actually select the referenced component.

New features include:

- Properties that reference a second component are red and the properties of that second component are green, by default.
- Events that reference a second component are red and the events of that second component are green, by default.
- Properties that are interfaces can be referenced inline.

Code Insight tools (All editions)

Code Insight has updated features for code completion and code parameters.

Code completion

When you manually (Ctrl+Space) invoke code completion on a blank statement line in the body of a routine, the pop-up list box now shows symbols from additional RTL units even if they are not used by the current unit.

New features:

- Filters out all interface method declarations that are referred to by property read or write clauses. The list box only shows properties and stand-alone methods declared in the interface type.
- Filters as you type in the Code editor. The text is automatically removed when you select an item.
- Works within a class declaration in the interface section with support for multiselect.
- Displays inherited and virtual methods, interface methods and properties.
- The pop-up window is resizable.
- Colors have been added to help distinguish different items. For example, procedures are teal and functions are dark blue. You can change default colors in the registry.
- Supports WM_xxx, CM_xxx, and CN_xxx message methods based on like named constants from all units in the **uses** clause.
- Abstract methods now appear in red in type declarations. All other Code Insight contexts show abstract methods using the normal procedure and function colors.

Code parameters

- If an item is a procedure or function with parameters, a "(" is included and a code parameter tooltip is displayed immediately.

New key mapping module (All editions)

The Visual Basic emulation key binding set has been added to the existing key mapping modules.

Packages in Project Manager (All editions)

Now all open package projects are displayed in the Project Manager view. These package project reference nodes can be used to assist tracking the active project as well as navigating to any open package, even when the package is not a member of the current project group.

File menu

The File menu has an updated File|New menu that displays six types of projects (Application, CLX Application, Data Module, Form, Frame, and Unit) along with Other, which brings up the New Items dialog box.

New Items dialog box

The New Items dialog box, or Object Repository, has three new pages with new wizards and data modules:

- WebSnap, which contains the WebSnap Application, WebSnap Data Module, and WebSnap Page Module.
- WebServices, which contains the Soap Server Application, Soap Server Data Module, and WebServices Importer.
- Corba, which contains the CORBA Client Application and CORBA Server Application. You can also access these wizards by choosing Tools|Regenerate CORBA IDL Files.

Internet toolbar (Enterprise edition)
The Internet toolbar is a new toolbar to help you create a WebSnap Web server application. Choose View|Toolbars|Internet to display the toolbar. The toolbar's icons access three new wizards, New WebSnap App, New WebSnap Page Module, and New WebSnap Data Module, and the External Editor. These buttons correspond with the most commonly used functions for creating a Web server application with WebSnap.

Component palette changes

- The Additional page has new action band components for maintaining action lists and building customized menus and toolbars, including TActionManager, TActionMainMenuBar, TActionToolBar, and TCustomizeDlg.
- For CLX applications, the Common Controls page substitutes the Win32 page. (Professional and Enterprise editions)
- A new WebSnap page has been added with components to build a Web server application with multiple modules. (Enterprise edition)
- The WebServices page has been added for writing multitier applications over a network or the Web. (Enterprise edition)
- Several changes have been made on the database Component palette pages. (Professional and Enterprise editions)
- A new COM+ page with the COMAdminCatalog component. (Professional and Enterprise editions)
- Three new Internet protocol pages have been added, including Indy Clients, Indy Servers, and Indy Misc. The Internet Direct (Indy) components are open source Internet protocols based on blocking sockets. (Professional and Enterprise editions)

Environment Options dialog box

The Tools|Environment Options dialog box has four new pages:

- Designer page—The options for the Form Designer have been moved from the Preferences page to the Designer page with one new option, Show extended control hints.
- Environment Variables page—You can view system environment variables, and create, edit, and delete user overrides to the system variables.
- Object Inspector page—You can set preferences for the Object Inspector using the new Properties dialog box, also accessible by right-clicking the Object Inspector and clicking Properties on the context menu.
- Internet page—You can set preferences for the file types and script options you want to use for your WebSnap applications. (Enterprise edition)

Directories dialog box

The Directories dialog box for editing items such as Library Path now displays invalid paths in gray. You can remove invalid paths with the Delete Invalid Paths button.

Context menu display

With an object selected on a form, a selected window, or with focus on any item in the IDE, you can now display associated context menus by pressing Alt-F10 or by pressing the Menu key (on Microsoft keyboards).

Writing design-time packages

You must add DesignIDE.dcp to your package's requires list.

New Internet features (Professional and Enterprise editions)

Support for WebServices (Enterprise edition)
New components and changes to the compiler let you write WebServices applications and support writing client applications that access WebServices using SOAP. These components and the architecture that supports them is described in Using Web Services.

Note: WebServices is a new and evolving technology. Because of this, the components that support this feature, including the interfaces, are subject to change.

Support for Web server applications (Professional and Enterprise editions)
Delphi 6 includes a host of new Internet features for building Web server applications, including changes to the existing WebBroker features and a new set of features called WebSnap. To figure out what type of application you want to build, see About WebBroker and WebSnap.

Apache Web Server (Professional and Enterprise editions)
Both WebSnap and WebBroker now support the Apache Web server in addition to ISAPI, CGI, and WinCGI.

Web Application Debugger (Professional and Enterprise editions)
The Web application debugger enables you to monitor HTTP requests, responses, and response time and to develop WebBroker and WebSnap applications without installing a commercial Web server. Debugging a Web server application can be difficult because your application needs to run in response to a message from a Web server. Therefore, the Web application debugger emulates this message.

Improved WebBroker features (Professional and Enterprise editions)
WebBroker now:

- Allows you to write Web server applications for CLX applications.
- Is backward compatible with Delphi 3, Delphi 4, and Delphi 5 Web modules, with some limitations. For example, old projects may need to add/change uses unit names. Multiple Web modules will not be supported in old applications.

New WebSnap features (Enterprise edition)
WebSnap is based on the Delphi 5 WebBroker features with a lot of new functionality, including:

- Multiple modules
- New wizards
- Server-side scripting
- New components
- Surface designers

To get familiar with building a WebSnap application, see the WebSnap tutorial.

Multiple Web modules
A single WebSnap application supports multiple Web modules, which partitions the application into units and allows multiple developers to work on the same project with fewer conflicts. Furthermore:

- Multiple instances support concurrent requests.
- Component references are supported between modules.
- Modules are automatically created to service a request or when referenced by another module.

New Web module wizards
With the Web server application wizard you can build an application with a customized Web moduleWeb application module or a Web page module, using the following four new components:

TWebAppPageModule
TWebAppDataModule
TWebPageModule

TWebDataModule

A Web data module creates a container for components that are shared across your Web application. It does not have an associated HTML template file.

A Web page module defines a new Web page in your application. It has an associated HTML template file and displays your script using the WebSnap surface designers. It also defines a logical page name, doesn't need to define a dispatch action to access the Web page, and manages the HTML template file by a virtual file system, displaying it in the Project Manager.

To access the Web module wizards, either choose File|New|Other, click the WebSnap tab, and double-click WebSnap application, or choose View|Toolbars|Internet and click the New WebSnap Application icon on the toolbar.

Server-side scripting

WebSnap includes server-side scripting, a simple scripting language designed for programming a Web page. It includes:

- A script that can access Delphi components.
- Active scripting engine support, such as VB or JavaScript.
- Mixed script and HTML.

New components

WebSnap comes with a host of new components, including dispatcher, adapter, page producer, session, and user list.

Dispatcher components

Dispatcher components automatically handle different types of requests for page content, HTML form submissions, and requests for dynamic images. New components include:

TPageDispatcher

TAdapterDispatcher

TWebDispatcher has a new OnException event: TCustomWebDispatcher.OnException

Adapter components

Adapters provide a means to define a scriptable interface to the business rules of your application. For example, TDataSetAdapter is used to make dataset components scriptable. New components include:

TAdapter

TApplicationAdapter

TPagedAdapter

TLoginFormAdapter

TApplicationAdapter

TEndUserAdapter

TEndUserSessionAdapter

TStringsValueList

TDataSetValueList

Page producer components

Page producers build complex data-driven forms and tables or use XSL to generate a page. New components include:

TAdapterPageProducer

TXSLPageProducer

Session components

Session components keep track of end users. New components include:

TSessionsService

User List components

User list components provide access to user names, passwords, and access rights. New components include:

TWebUserList

WebSnap surface designers

When you are scripting a Web server application with WebSnap, you can build Web pages and view the results at design time on the Code editor on the following Code editor pages: HTML Script, HTML Result, Preview, XML Tree, and XSL Tree.

XML support (Enterprise edition)

XML Data Binding wizard

This new wizard creates Delphi code that provides a very simple way to access and update XML data files. The generated code presents a natural and intuitive property-based programming model that makes working with XML data feel just like programming with a Delphi component. XML data bindings replace the complex coding required to use the XML Document Object Model (DOM) interfaces, and are ideally suited to applications that manipulate validated XML documents.

XML document programming

The XML data bindings are based on the new TXMLDocument component and its associated interfaces IXMLDocument/IXMLNode. This component provides a simplified version of DOM interfaces for working with XML data. Just drop TXMLDocument on your form, set the FileName property, set Active to true, and you have instant access to all of your XML data.

Cross platform/vendor-independent DOM programming

At the lowest level in the XML document programming support is the new xmldom.pas interface unit that provides a cross platform and vendor independent set of interfaces for programming with the W3C DOM Level 2 specification. Designed with an open architecture, the interfaces are easily integrated with existing DOM-based XML solutions.

Using XML in database applications

A new set of components enables you to integrate XML documents into the product's database architecture. The components make use of transformation files generated by XML Mapper, a design-time tool for defining mappings between an arbitrary XML document and a client dataset's data packet.

New compiler features

Variants

All the variant handling routines were moved from the System unit to a new unit called Variants. Although you can still use variant types without including the Variants unit in your applications, the RTL functions manipulating variants are in the Variants unit. To use these functions, you'll need to include the Variants unit in the uses statement of your application.

You can now define custom data types for variants. See Custom variant support.

Variants now support Int64.

Enumerated types

Enumerations may now be assigned a specific value.

For example:

```
type
  TInfo = (iZero, iOne, iTwo, iFour = 4);
```

Consts unit changes

Consts.pas was broken up into two files: Consts.pas and RTLConsts.pas.

New compiler directives

There are new library, deprecated, and platform directives, documented in the Object Pascal Language Guide.

Microsoft increasingly relies on PE (portable executable) header flags to allow an application to indicate compatibility with OS services or request advanced OS services. The Delphi 6 compiler now supports two new directives that provide powerful options for tuning your applications on high-end NT systems:

```
{$SetPEFlags <integer expression>}
```

and

```
{$SetPEOptFlags <integer expression>}
```

These directives are for advanced developers only. See PE (portable executable) header flags.

Conditional directives

If you are writing for a Windows platform, you can use the symbol MSWINDOWS to detect the presence of Windows rather than Linux. For example:

```
{$IFDEF MSWINDOWS}
  Get Desperate;
{$ENDIF}
```

$If directive

There is a new $IF directive with constant expression evaluation. For example:

```
{$IF Defined(WIN32) and (SomeConst > 12.0) }
...
{$ENDIF}
```

Pascal constant identifiers can be evaluated in $IF directives. Existence of a conditional define symbol ($IFDEF) can be tested with the Defined() intrinsic function, and existence of a Pascal constant identifier symbol can be tested with the Declared() intrinsic function.

For example:

```
{$IF  Defined(WIN32) and Declared(MyConst)}
...
{$ENDIF}
```

$ALIGN field

The $ALIGN field has new options: {$A1}, {$A2}, {$A4}, and {$A8}.

New built-in assembler

The product now has a completely new built-in assembler with:

- New directives, VMTOFFSET and DMTINDEX
- New instruction support: MMX, SIMD, Enhanced MMX, and Intel SSE for the Pentium Pro, Pentium III, and Pentium 4 CPUs; and AMD Enhanced 3D for the AMD K7 CPUs.
- New support for DQ (define quadword data) pseudo-opcode.

New default setting for writeable constants

The default setting for $WRITEABLECONST ($J) compiler directive has changed from ON to OFF. This means that you must explicitly turn this compiler flag on before you can write to typed const values.

File type support

For greater cross-platform support, the compiler can handle Linux-style text files. That is, text files whose lines end only with the linefeed character (ASCII 10) instead of the carriage return (ASCII 13) and linefeed (ASCII 10) used in Windows.

Overload resolution changes

The process by which the Delphi compiler decides which overloaded function best matches a particular argument list has been updated to support the following situations that the compiler previously considered ambiguous:

- The Delphi 6 compiler can now distinguish between overloaded functions that contain AnsiString/PChar and WideString/WideChar parameters in the same parameter position.
- Variants can now be parameters in overloaded function declarations. A Variant is considered more general than any simple type.
- Object instances passed to overloads with an object version and an interface version are no longer ambiguous.
- Nil is now accepted as a valid value for an interface-type parameter in an overloaded function.

See Overloading procedures and functions.

New COM/Active X features (Professional and Enterprise editions)

Registration/installation of COM configuration attributes
You can now set COM+ attributes to new COM objects.

Event Object Wizard
A new COM+ event wizard lets you create COM+ event objects. (You must still add code manually to fire events or client code to respond to events.)

Implementing existing interfaces
You can now use the COM object wizard to generate a server object for an arbitrary interface that is in a type library registered on your system. Previously, that wizard always implemented a newly created interface (descending from IUnknown). Now, the COM object wizard also lets you select an interface from any registered type library. The COM object wizard creates the type library information for a CoClass to implement that interface, as well as an implementation class with skeletal methods for you to fill in to complete the implementation. The implementation class inherits an implementation of IUnknown and IDispatch methods.

Transactional Objects (MTS Wizard replacement)
Transactional objects can now be created using the Professional edition. Previously, MTS support was limited to the Enterprise edition.

Dual MTS/COM+ support for transactional objects
The MTS object wizard has been replaced by a transactional object wizard, which creates objects that can be used with either COM+ or MTS.

New database features (Professional and Enterprise editions)

Delphi 6 includes a new data access mechanism, *dbExpress*, that provides extremely fast, easy-to-deploy access to SQL database servers using a framework that makes it easy to write third-party database drivers.

Datasets now support two new field types.

A number of new components have been added to make it easier to work with client datasets, both for two-tier and multi-tier database applications.

dbExpress

dbExpress is a set of lightweight database drivers that provide fast access to SQL database servers. For each supported database, dbExpress provides a driver that adapts the server-specific software to a set of uniform dbExpress interfaces. When you deploy your application, you need to include either a single DLL (the server-specific driver) with the application files you build with optional, additional files that let you load connection information at runtime. Or, you can deploy your application as a standalone .EXE.

Depending on your version, the following dbExpress drivers may be included with Delphi:

Driver	File name
InterBase driver	DBEXPINT.DLL
DB2 driver	DBEXPDB2.DLL
Oracle driver	DBEXPORA.DLL
MySQL driver	DBEXPMYS.DLL

The main components, which are wrappers around *dbExpress* drivers,include:

- TSQLConnection.
- TSQLDataSet, a unidirectional dataset.
- TSQLQuery, TSQLStoredProc, and TSQLTable, which are unidirectional datasets that are compatible with existing elements if you want to port an application.

For more information about working with these components, see Using unidirectional datasets.

The datasets that use dbExpress are called unidirectional datasets, because dbExpress implements only a unidirectional cursor for accessing records. Unidirectional datasets do not buffer data in memory, which makes them faster and less resource-intensive than other types of dataset, but introduces several limitations. For example, you can't connect a unidirectional dataset to a data-aware grid, because there are no buffered records. Many of the capabilities introduced by TDataSet are either unimplemented in unidirectional datasets, or cause them to raise exceptions. Despite the limitations, unidirectional datasets are a powerful way to access data. They are fast, and very simple to use and deploy.

To edit data from unidirectional datasets, connect them to a client dataset. The connection between the client dataset and the unidirectional dataset is provided by a dataset provider.

New field types

A new field type, TFMTBCDField, has been added. This represents a true binary-coded decimal, as opposed to the existing TBCDField type, which converted BCD values to the Currency type. You can specify a TFMTBCDField as a persistent field for a BCD field from any dataset. With dbExpress datasets, dynamic field components of type TFMTBCDField are created when using TBCDField would result in a loss of precision. (Eventually, this may be expanded to include BDE, ADO, and/or IBX datasets as well).

Note: In support of TFMTBCDField, the global CurrToBCD and BCDToCurr routines have moved to the FMTBCD unit (from the db unit).

Another new field type, TSQLTimeStampField supports the date/time representation used in *dbExpress* drivers.

Client dataset enhancements

Three new client datasets have been added that include a built-in provider and source dataset. These datasets are intended to simplify the process of using a provider and client dataset to cache updates in very simple applications. They are:

- TBDEClientDataSet
- TSQLClientDataSet
- TIBClientDataSet

A new common base class (TCustomClientDataset) now exists for both TClientDataSet and these new datasets.

Note: These components are only intended for simple applications. When the application uses master/detail relationships or needs to refetch data from the server multiple times, performance is much better when using an external provider and client dataset.

TClientDataSet has several new properties:

- ConnectionBroker lets you add an extra layer of indirection to the specification of a connection component. This is especially useful when you want to use one type of connection at design time and another when you deploy your application. If you have several client datasets in an application that all use the same connection broker, then you can change the component that connects them all to the application server by changing a single property (the Connection property of the connection broker) rather than having to change the RemoteServer property for every client dataset in the application.
- DisableStringTrim lets you control how the client dataset handles spaces in values that users enter into fields.
- XMLData gives you access to the client dataset's data packet in XML format.

A new set of components lets you convert between client datasets and arbitrary XML documents. These are described in Using XML in database applications.

TUpdateSQL has been enhanced so that you can now use multiple update objects when caching updates using a client dataset. The TUpdateSQL.DataSet property has changed from requiring a TBDEDataSet to only requiring TDataSet. When using a client dataset and multiple update objects, you should set this to the DeltaDS parameter of the provider's BeforeUpdateRecord event handler. In this case, TUpdateSQL can't infer the database name and session from its DataSet property and you must also set the new DatabaseName and SessionName properties. This process is described in Using Multiple Update Objects.

TUpdateSQL works the same as before when there is only a single update object or when using the BDE-enabled dataset to cache updates.

Support for multi-tiered applications

Two new connection components are available that let you work more flexibly with client datasets in multi-tiered applications. These are:

- TSharedConnection, which lets the client application form several connections to multiple remote data modules in a single application server. By using TSharedConnection components, all the connections to the remote data modules use a single shared connection to the application server, allowing the server application to recognize them as all originating from the same client.
- TLocalConnection, which acts like a connection component for local providers (in the same application as the client dataset). By using TLocalConnection, you can make explicit use of the IAppServer interface, and it can simplify the process of later scaling up to the use of a remote provider on an application server.

In addition, a new factory object, TPacketInterceptFactory, makes it easier to intercept messages between the client application and the application server when using socket connections. By using TPacketInterceptFactory, the intercept object is automatically registered so that you can assign it to the socket connection component using a drop-down list in the Object Inspector.

Component palette changes

The component palette has been changed to reflect the growing variety of options available when working with databases and Internet applications.

- The component palette has been reorganized to emphasize that the Borland Database Engine is

one choice among many, rather than the primary data access mechanism. The BDE-based components have been moved to a new BDE page of the Component palette.

- The InterBaseExpress components have been updated, including enhanced event support and support for InterBase generators that auto-generate field values. A new InterBase Admin page has been added.
- A new dbExpress page has been added to house the new dbExpress components described previously.
- The Midas page has been removed, as has use of the term MIDAS. The client dataset and dataset provider that used to be on the Midas page has moved to the Data Access page. The connection components for connecting to an application server and the simple object broker have moved to the new DataSnap page.
- The new DataSnap page now holds the components used for connecting client datasets to an application server. These include the connection components and simple object broker, that used to be on the Midas page. In addition, the DataSnap page holds three new components: TConnectionBroker, which acts as an intermediate component between a client dataset and a connection component, TSharedConnection, which connects to an application server that is partitioned into multiple remote data modules, and TLocalConnection, which represents the connection to providers that are in the same application as a client dataset.

New CORBA features (Enterprise edition)

Some versions of Delphi come with an IDL2PAS compiler for writing CORBA applications. This IDL2PAS compiler has been enhanced to support writing CORBA servers (generating skeleton code) as well as clients (generating stub code). You can launch this IDL2PAS compiler from within the IDE by choosing Tools|Regenerate CORBA IDL files or by choosing File|New|Other and clicking the CORBA page. The compiler is useful for creating clients and servers, and maintaining existing CORBA projects where you make a change to the IDL file and just want to refresh the project files.

It is highly recommended that you use the new IDL2PAS compiler for CORBA applications, rather than using the older CORBA support that was integrated with Delphi's support for COM applications. When using the new IDL2PAS compiler, you should use the corba.pas and orbpas30.pas or orbpas40.pas units to interact with the ORB rather than the older corbaobj.pas and orbpas.pas units. In addition, two new units, cosevent.pas and cosnaming.pas wrap the event and naming services, respectively.

The documentation for the CORBA support has not been integrated into the main Delphi documentation. Instead, the IDL2PAS compiler is documented in Doc\IDL2PAS\index.htm.

New actions features (Professional and Enterprise editions)

Action bands

Actions are easier to work with, simplifying the process of developing your user interface by using a set of new tools collectively known as ActionBands. You can organize actions and images and add them to customizable, Microsoft Office-style menus and toolbars. These tools are accessed from the Additional page of the Component palette and include:

- TActionManager—The Action Manager manages action lists to organize customized and standard actions.
- TActionMainMenuBar and TActionToolBar—The action band menu and toolbar are customized menus and toolbars onto which you can drag and drop actions, or commands, from the Action Manager editor. You can drop a category of actions (you get all of the actions included on a submenu) or a specific action onto a menu or toolbar.
- TCustomizeDlg—The Customize dialog box works in the same way as the Action Manager editor, but is provided for your users to modify the contents of menus and toolbars at runtime. TCustomizeDlg component works in conjunction with TActionManager: you can either set the ActionManager property to the ActionManager component, or add a standard action (TCustomizeActionBars) to the Action Manager editor and to an action band menu.

For more information on using these tools, see Organizing actions for toolbars and menus and Creating toolbars and menus. For CLX applications, use the ActionList to organize your actions.

Delphi includes new standard actions that you can add to your menu and toolbar commands. New predefined action classes include Format, File, Search, Tab, List, Dialog, Internet, and Tools.

You can also set up your menus to hide or display actions based on their frequency of usage. You do this using the HideUnused property of the various TActionClient objects, which places less frequently used menu items on a submenu accessible by pointing to a double arrow button on a menu or toolbar. For more information, see Hiding unused items and categories in action bands.

New standard actions

The following standard actions have been added to the VCL:

Format actions

- TRichEditBold
- TRichEditItalic
- TRichEditUnderline
- TRichEditStrikeOut
- TRichEditBullets
- TRichEditAlignLeft
- TRichEditAlignRight
- TRichEditAlignCenter

Help actions

- THelpContextAction

File actions

- TFileOpen
- TFileSaveAs
- TFilePrintSetup
- TFileRun
- TFileExit

Search actions

- TSearchFind
- TSearchFindFirst
- TSearchReplace
- TSearchFindNext

Tab (page control) actions

- TPreviousTab
- TNextTab

List actions

- TListControlCopySelection
- TListControlDeleteSelection
- TListControlSelectAll
- TListControlClearSelection
- TListControlMoveSelection
- TStaticListAction
- TVirtualListAction

Dialog actions

- TOpenPicture
- TSavePicture
- TColorSelect
- TFontEdit
- TPrintDlg

Internet actions

- TBrowseURL
- TDownLoadURL
- TSendMail

Tools actions

- TCustomizeActionBars

Enhancements to action classes

TCustomAction has been enhanced to include the following new properties:

- GroupIndex
- SecondaryShortCuts
- HelpKeyword
- HelpType
- AutoCheck

TCustomActionList now has a State property that lets you temporarily turn off all the actions in the list.

New VCL units and features (All editions)

New components:

TLabeledEdit
TLabeledEdit is an addition to the ExtCtrls.pas unit and demonstrates using sub-components. It is an Edit control that has a label attached to it. The Label appears as a property of the control.

TValueListEditor
The TValueListEditor is a custom grid used for editing TStrings that contain key/value pairs. It works similarly to the Object Inspector.

TComboBoxEx
TComboBoxEx is new combo box control that allows images to appear next to the items in the list.

TColorBox
TColorBox is new combo box control for selecting colors.

Improved features include:
Most of the windowed controls now publish the following Bevel properties:

- BevelEdges
- BevelInner
- BevelOuter
- BevelKind
- BevelWidth

Subcomponents
Components can now own other components that create subcomponents. For example, a component can have a property that is a component reference and that can be either internal (a subcomponent) or external (a normal component reference). If the reference is internal, the subcomponent is not owned by the form but by the component on the form. This means that components can now publish the properties of the subcomponent and they will be streamed correctly. Additionally, the Object Inspector has been modified to allow you to view the properties of component references inline (for example, the Font property). To create a component that has a sub-component requires a call to TComponent.SetSubComponent.

Publishable interface properties
Interface properties (properties whose type is an interface) can now be published if and only if the implementer of the property is a streamable component.

This means you can now see properties that take an interface in the Object Inspector, which will provide a drop-down list of components that support the interface.

Unit additions and changes

CheckLst.pas
TCheckListBox now publishes several new properties, including AutoComplete, HeaderColor, and HeaderBackgroundColor.

Classes.pas
TList has a new Assign method that not only copies but also allows for primitive set operations.

TCollection has two new protected methods that are used to allow descendants of TCollection to enhance the process of adding and deleting items. The two new methods are Added and Deleting, neither of which have a default implementation. There are no OnAdded and OnDeleting events, to keep TCollection small. However, descendant classes can easily add these events. In addition, TCollection has a new Owner property to make it easier to identify the owner.

TStringList has a new CaseSensitive property that lets you control whether string list operations

(sorting, string matching) are case sensitive.

TDataModule was moved here from the Forms unit to remove dependencies on visual controls. This lets you write much smaller server applications that include no user interface.

TThread has a new FatalException property that identifies any exception that stops the thread function from a normal completion.

TStream overloaded the Seek function to allow Int64 values to be used to identify positions. Descendant classes can override one or the other overload, but should not override both.

TInterfacedPersistent is a new base class for persistent objects that are not components but that implement interfaces.

ComCtrls.pas

TTreeView—CreateTreeNodes has been added.

- Generalized the creation of nodes and made an event so that simple tree users won't have to create a descendent just to override the node class. Also AddNode has been changed so you can now pass in the node (of whatever class) you want added.
- Added OnAddition event, which occurs when nodes are added.
- The API for sorting TreeViews has been augmented and simplified. You can now recursively sort subtrees, and non-recursively sort the top-level nodes. The TCustomTreeView and TTreeNode classes now offer more uniform definitions for AlphaSort and CustomSort, and these methods have been added to TTreeNodes. All changes are backward compatible with the previous version.
- Added MultiSelect with four properties and eight methods.

TListView—CreateListItems has been added, which matches TTreeView's CreateTreeNodes.

TStatusBar—The size grip now appears even if the status panel is not directly parented by its form. As long as the lower right corner of the status bar is in the lower right corner of the form then the size grip will appear.

TDateTimePicker has a new Format property that controls the format of the date values using standard date/time formatting strings.

THeaderControl has a number of new properties and events to support column dragging. A new HotTrack property allows header sections to be highlighted when the user pauses with the mouse.

TToolBar now has a Menu property that fills the toolbar with buttons corresponding to the items in a menu. A number of new events respond when the user customizes the toolbar using a customize dialog.

Contnrs.pas

Last and First have been added to TObjectList, TComponentList, and TClassList. These are typecast functions.

TStack, TQueue, TObjectStack, and TObjectQueue's Push is now a function that simply returns the item pushed on to the stack. Think of it as a Push/Peek. This is very useful when pushing things that are created when pushed.

TBucketList and TObjectBucketList are simple hash tables.

Controls.pas

TCustomListControl is a new common base class for controls that represent a list of items (combo boxes, list boxes, and list views). It introduces a number of new methods for manipulating the lists that are then inherited by all descendants.

TDragObjectEx, TDragControlObjectEx, and TDragDockObjectEx, are three new drag objects that are automatically freed at the end of drag operations. They correspond to TDragObject, TDragControlObject, and TDragDockObject, differing only in that the older objects are not freed at the end of a drag operation.

TControl has two new methods: ClientToParent and ParentToClient, that let you map your pixels to one of the parents or children. They work very similarly to ClientToScreen and ScreenToClient.

TWinControl has a new overload of the PaintTo method that takes a canvas instead of an HDC.

TModalResult has been moved from Forms.pas. In addition, the following support functions have

been added:

```
function IsPositiveResult(const AModalResult: TModalResult): Boolean;
function IsNegativeResult(const AModalResult: TModalResult): Boolean;
function IsAbortResult(const AModalResult: TModalResult): Boolean;
function IsAnAllResult(const AModalResult: TModalResult): Boolean;
function StripAllFromResult(const AModalResult: TModalResult):
TModalResult;
```

DbCtrls.pas

TDBLookupListBox and TDBLookupComboBox have a new NullValueKey property that lets users assign blank (NULL) values.

TDBComboBox now has AutoComplete and AutoDropDown properties.

TDBListBox also has a new AutoComplete property.

TOpenDialog now has an OptionsEx property that expands your control over open and save dialogs.

ExtCtrls.pas

TImage—A Proportional property has been added which maintains the aspect ratio of the image regardless of the size of the TImage control.

Forms.pas

TApplication (and TApplicationEvents) has a new OnSettingChange event that lets you respond to changes in system-wide settings.

TForm now supports layered forms with the new AlphaBlend, AlphaBlendValue, TransparentColor, and TransparentColorValue properties.

TScreen has a new set of properties for getting the work area of the desktop (WorkAreaRect, WorkAreaTop, WorkAreaLeft, WorkAreaHeight, and WorkAreaWidth). In addition, a set of new methods let you locate which monitor best matches a point, rectangle, or window.

TMonitor expands the multi-monitor support by indicating which monitor is the primary monitor, and providing WorkareaRect and BoundsRect properties.

AutoDragDocking support has been added. This allows you to turn off auto-docking for your application. Additionally a flag has been added to Delphi's options dialog that allows you to set this property.

TModalResult has been moved to Controls.pas.

Graphics.pas

TFontRecall, TPenRecall and TBrushRecall have been added. These allow quick saving and restoring of fonts, pens, and brushes. They descend from TRecall (from Classes) which works with TPersistent classes in general.

The system colors have been have sorted to make things easier to find.

Four colors have been added to the standard sixteen:

- clMoneyGreen
- clSkyBlue
- clCream
- clMedGray

ImgList.pas

Overloads have been added for the Draw, DrawOverlay, and GetIcon methods, which let you override the image list property settings.

IniFiles.pas

Ini files now support reading and writing binary data using a stream.

TMemIniFile now lets you control whether strings are treated in a case-sensitive manner.

THashedStringList is a new TStrings descendant that uses an internal hash table to improve performance.

Masks.pas
EditMask and Text now use custom types so their property editors are more useful.

Menus.pas
TMenuItem has a new AutoCheck property that checks or unchecks menu items automatically when the user selects them.

Registry.pas
TRegistry now supports reading and writing binary data using a stream.

StdCtrls.pas
OnCloseUp and OnSelect have been added to TCustomComboBox (and in turn TComboBox). OnCloseUp fires when the combo box's drop-down list closes (think of it as the opposite of OnDropDown). OnSelect fires when something is selected from the combo box's drop-down list (or when the contents of the combo box are changed by scrolling its contents up and down). Combo boxes now have an AutoComplete property which is true by default.

TListBox now supports two new styles: lbVirtual and lbVirtualOwnerDraw. These styles are for virtual list boxes, which do not store their items. Rather, you indicate the number of items by setting the Count property, and then supply items (and their associated objects) using the new OnData, OnDataFind, and OnDataObject events.

TypInfo.pas
It is now safe to call GetPropInfo with an object that does not have any RTTI information. It simply returns nil.

FreeAndNilProperties has been added. This will take any RTTI-enabled object and free and nil out each of its object properties. Note that it will also clear any objects this object may have property references to, so nil those out first.

New RTL units and features (All editions)

Some functions have been moved from other units into System, while many System functions have moved to the new Variants unit. Many new overloaded versions of RTL functions have been added to support WideString parameters in addition to AnsiString parameters (SysUtils), such as Trim and WideFormat.

Variants.pas
This new unit contains many utilities for working with Variants. Much of this code was moved from the system unit, so that now, if your code uses Variants, you must add Variants to you uses clause. This unit also contains a number of new Variant support routines, as well as support for the new Custom Variant types.

To support cross-platform development, the utilities in the Variants unit no longer make calls directly into the Windows API. Instead, a new unit, VarUtils.pas, contains low-level routines that provide a platform-neutral basis for the routines in Variants.pas. You have to be careful using the generic code in VarUtils.pas because it is not exactly like the Windows code and if you mix Windows calls and generic code you will run into problems.

ConvUtils.pas
A collection of routines for converting between measures has been added.

StdConvs.pas
A set of global variables for use with the routines in ConvUtils.pas.

DateUtils.pas
A collection of date and time functions has been added.

StrUtils.pas
This new unit contains string functions in addition to the ones in SysUtils.pas.

FMTBCD.pas
This new unit contains utilities for working with binary-coded decimal (BCD) values.

Math.pas
Const has been added to the extended parameters to speed things up. There are also many new constants and functions.

System.pas
The System unit has many new routines, most of which support cross-platform development and conversion between the different character encoding systems that are used on Windows and Linux. The routines that support Variants have been moved out to the new Variants unit.

IInterface has been added for use with other than COM interfaces.

SysUtils.pas
The SysUtils unit has been upgraded to support the cross-platform features of CLX. Some services that were previously provided by the native Windows API are now provided by SysUtils routines. However, many of these routines appear in Linux-specific areas of the source file and are not compiled into the Windows binary.

Custom Variant support (All editions)

You can now define custom data types for Variants. This introduces operator overloading while the type is assigned to the Variant. To create a new Variant type, descend from the class, TCustomVariantType (or one of its descendants, TInvokeableVariantType or TPublishableVariantType) and instantiate your new Variant type.

See Defining custom Variants for more information on how to create custom Variants.

Two new units provide examples of custom Variants:

- The VarCmplx unit implements a custom Variant for complex numbers. The Variant type supports direct manipulation using the addition, subtraction, multiplication, division (but not integer division), and negation operators. It supports 5 properties: Real, Imaginary, Radius, Theta, and FixedTheta. It can be cast to and from integer types, floating point types, string types, TDateTime values, and boolean values. In addition, the VarCmplx unit implements a number of global functions for operating on complex Variants.
- The VarConv unit implements a custom Variant for measurements such as those used in the ConvUtils unit. Convert custom Variants support addition, subtraction, multiplication, and division between Convert custom Variants and numbers, or between two Convert custom Variants (except that you can't multiply two Convert custom Variants that use different units). The Convert Variant type automatically adjusts the units when you perform these operations. You can cast Convert Variants to OleStr, String, and Double. They also support Value, Type, TypeName, Family, and FamilyName, and As<Unit> properties for obtaining the numerical value, the unit type, the name of the unit, the measurement family to which the unit belongs, its name, and the value when converted to another unit in the same family.

Cross-platform development (Professional and Enterprise editions)

Delphi 6 ships with the Borland Component Library for Cross-Platform (CLX), a class library similar to the VCL that can run on both Windows and Linux platforms. Many of the CLX objects are named the same as VCL objects and include many of the same properties, methods, and events. You can use CLX with Delphi 6 to develop applications that can be compiled under either Windows or Linux to run on Linux.

CLX differences

There are differences due to the operating system and features that relate to technologies that are limited to Windows:

- The CLX class library between VCL and are similar although CLX's TWidgetControl replaces the VCL's TWinControl. Thus, other components, such as TScrollingWidget, have corresponding names.
- The IDE uses a smaller subset of objects in the Object Repository and on the Component palette pages.
- All of the variant/safe array code that was in System is in two new units: Variants.pas and VarUtils.pas
- You can define custom data types for variants. This introduces operator overloading while the type is assigned to the variant. To create a new variant type, descend from the class, TCustomVariantType, and instantiate your new variant type.
- Application-wide "styles" can be used in addition to the OwnerDraw properties. You can use the TApplication.Style property to specify the look and feel of an application's graphical elements.
- Most strings for controls in CLX are wide strings whereas in VCL they are ANSI strings. Typecasting a wide string to PChar or String to PWideChar won't work and will cause the compiler to issue a "Suspicious typecast " warning.
- Linux does not use a registry to store configuration information. Instead, you use text configuration files and environment variables.

Creating a cross-platform application

To create a cross-platform application, choose File|New|CLX Application. The Component palette changes dynamically to show the objects that are available for use in Windows CLX applications.

There are some Windows-specific features that do not port to the Linux environment.

Improved translation tools (Enterprise edition)

The term Translation Tools, which includes the Translation Manager, Resource DLL Wizard, and Translation Repository, has replaced the term Integrated Translation Environment (ITE). The Translation Manager is now a standalone executable that can be used outside the IDE. When used externally, it is called the External Translation Manager (ETM) (etm60.exe), and can be sent to translators without their needing to install Delphi.

The Translation Manager and ETM dialog boxes have some added functionality:

- A form viewer so that users can view and visually resize the form as they make translations.
- Three new Environment, Files, and Workspace tabs to organize your project.
- Additional menu and toolbar buttons for File, Project, and Tools commands. The commands vary depending on whether you are running the Translation Manager internally or externally.

Deployment changes (All editions)

In Delphi 5, users deployed runtime packages by distributing the appropriate packages (.BPLs) with their application, using InstallShield to automatically include needed libraries.

Delphi 6 ships with InstallShield Express 3.0, which is based on Windows Installer (MSI) technology and uses merge modules to deploy your application. Merge modules are MSI components that contain files and logic necessary to install the runtime libraries. Delphi's libraries have interdependencies which Delphi's merge modules are designed to resolve.

To deploy applications with InstallShield 3.0 and to display a list of the library dependencies for each Delphi merge module, see deploying applications and Merge modules.

Help system extensibility (All editions)

Delphi now allows you to pass Help requests to Help viewers other than the standard Windows Help viewer, enabling you to write Help applications for both Windows- and Linux-based applications. New features include:

- New interfaces that communicate between your application and Help viewers. Defined in HelpIntfs.pas, these interfaces include IExtendedViewer, ISpecialHelpWinViewer, IHelpManager, IHelpSystem, IHelpSelector, and ICustomHelpViewer.
- The Help Manager, which maintains a list of registered viewers and passes requests to them.
- The VCL provides an implementation of ICustomHelpViewer designed for talking to WinHelp.

Note: Third-party component developers wishing to provide Help using the standard Windows Help engine must now provide a list of "ALinks" used in the compiled Help file. The list must be given an .ALS file extension and copied into the directory specified by the Delphi registry entry Help| WinHelpPath. ALink lists can be created from compiled Help files using the Report feature in the Help Compiler Workshop (installed into the Tools folder of your Delphi/Help directory). See the file DELPHI6.ALS in your /Help directory for an example of ALS list preparation.

Upgrading to Delphi 6

When you load a Delphi 5 or earlier project into Delphi 6, it is automatically updated. The following topics describe changes that could potentially impact existing Delphi projects:

- IDE issues
- Automatic package name updates
- Compatibility issues

Refer also to the What's New for information on additional features that you may want to incorporate into your applications.

IDE issues

When you open your project for the first time in Delphi 6, it will not look the way it did in previous versions of Delphi. There have been many changes to the IDE, including changes to the window layout and the component palette. For a more complete summary of these changes, refer to the New IDE Features section of the What's New document.

Automatic package name updates

As in prior Delphi upgrades, the names of many packages have changed. For example, "vclide50" is now "vclide60." Past releases of Delphi performed automatic package name updates, and this release is no different. The mechanism for the updates has changed, however. It now relies on a list of changed files (found in the file Delphi.upg in the Bin directory) instead of simply changing suffixes.

In the past, when Delphi detected an error that could be related to a package name change, it searched the package names for obselete suffixes and automatically replaced them with the correct suffix. For example, Delphi 5 would have automatically updated a reference to "package40" (if such a package existed) so that it read "package50" instead. Delphi 6 performs this task by searching a list of obselete package names and replacing them as indicated by the list. The list is contained in the file Delphi.upg, which can be found in the Bin directory.

Users can examine and change the contents of the file using a text editor (including the Delphi IDE's editor). This may be a useful feature for component designers, who might want to modify their components for Delphi 6 and keep the components in a different package than they used previously.

Compatibility issues

Following are general compatibility issues that may affect your Delphi applications:

Provider, client dataset events affected by VCL hierarchy change

Code change required for default Database Login dialog

Potential binary form file incompatibilities

Change in writeable constants

Unary negation of Cardinal type

DsgnIntf renamed and related changes

Component editor changes

TDesignWindow changes

VCL package changes

OpenGL interface unit moved to rtl.dcp

Types moved from HTTPApp.pas to HTTPProd.pas

Search unit removed, SearchBuf moved and changed

Some specific topics in the What's New document may have compatibility consequences for your application, such as:

- New compiler features
- New VCL features, particularly the "Unit additions and changes" section

Provider, client dataset events affected by VCL hierarchy change

Compatibility issues

The introduction of TCustomClientDataSet requires changes to event handlers in Delphi 5 and earlier code.

Six events in five types are affected by the change to DBCLIENT.PAS. They are:

Type	Event/change
TResolverErrorEvent	Affects the provider's *OnUpDateError* event.
TBeforeUpdateRecordEvent	Affects the provider's *BeforeUpdateRecord* event.
TAfterUpdateRecordEvent	Affects the provider's *AfterUpdateRecord* event.
TProviderDataEvent	Affects the provider's *OnGetData* and *OnUpdateData* events.
TReconcileErrorEvent	Affects the client dataset's *OnReconcileError* event

In event handlers using the events noted above, you must replace TClientDataSet with TCustomClientDataSet.

Code change required for default Database Login dialog

Compatibility issues

Previously, setting the LoginPrompt property of a connection component (such as TDatabase, TADOConnection, or TDCOMConnection) caused a default login dialog to appear. This is no longer the case unless you add DBLogDlg to your **uses** clause. Applications that depend on the default login dialog must be edited to add DBLogDlg to the **uses** clause or they will not prompt for user name and password.

Potential binary form file incompatibilities

Compatibility issues

In the past, binary form files (or DFM files) created with newer versions of Delphi could be read by older versions. This is no longer true in Delphi 6; some binary form files may be read incorrectly because of the way that Delphi 6 performs internal string streaming. In the past, streaming was performed assuming a locale specific character set. Now streaming assumes that the character set is UTF-8. As a consequence, if there are characters with a code greater than 127 (such as the copyright symbol ©) in a Delphi 6 binary form file, that file cannot be read by older versions of Delphi.

If you intend to use a Delphi 6 form file (including older form files imported into and modified with Delphi 6) in an older version of Delphi, the file should be saved in text format instead of binary format.

Change in writeable constants

Compatibility issues

The $WRITEABLECONST compiler switch (aka $J) now has a default state of OFF, which will prevent Delphi projects from having writeable constants. Writeable constants refers to the use of a typed constant as a variable modifiable at runtime. Here is an example:

```
const
   foo: Integer = 12;
begin
   foo := 14;
end.
```

In prior releases of Delphi, this was an acceptable construct; constants weren't really constant. With $WRITEABLECONST OFF, this code will now produce a compile error on the assignment to the foo variable in the begin..end block. To fix it, simply change the const declaration to a var declaration.

You may have code that uses a typed const as an initialized local variable with global lifetime, like this:

```
procedure MyProc;
const
   somedata: Integer = 12;
begin
   Inc(somedata, 3);
end;
```

You will need to move the local const out of the procedure and declare it for what it is: a global variable. After making this change, the code segment above becomes:

```
var
   somedata: Integer = 12;
procedure MyProc;
begin
   Inc(somedata, 3);
end;
```

Code that relies heavily on the typed const quirk (such as the ActiveX control wrapper generator) can insert a {$WRITEABLECONST ON} directive in the source file as a quick fix. This practice is forbidden in the RTL, VCL, CLX, and DB source code and discouraged in the IDE sources, but acceptable for fringe units such as ActiveX control wrappers.

In general, you should note that the phrase "writeable constant" is an oxymoron. Prior versions of Delphi allowed them by default to maintain compatibility with an older 16-bit compiler, which is no longer important for most Delphi developers. Use good programming practice; avoid writeable constants.

Unary negation of Cardinal type

Compatibility issues

In the past, Delphi handled unary negation of Cardinal type numbers using 32 bit operations, which could lead to some odd results. Here is an example of code which uses unary negation:

```
var
   c: Cardinal;
   i: Int64;
begin
   c := 4294967294;
   i := -c;
   WriteLn(i);
end;
```

In previous versions of Delphi, the value of i displayed would be 2. This is obviously incorrect behavior for this case. In Delphi 6, the unary negation is handled after promoting the Cardinal type to a 64 bit signed type, so the final value of i displayed is -4294967294.

It is possible that existing code may rely on the incorrect behavior of unary negation. Delphi users should be aware of this new behavior. It may be worth your time to check your code for instances of unary negation of Cardinal variables, and make sure that your application responds to the new behavior appropriately.

DsgnIntf renamed and related changes

Compatibility issues

References to DsgnIntf in your project should be changed to the new Delphi 6 name, DesignIntf. You may also need to add DesignEditors, VCLEditors and RTLConsts to your **uses** clause. You will also need to add designide to your package's **requires** list. References to dsnide50 should probably also be changed to designide if that isn't changed automatically by Delphi.

Any runtime packages that use IDesigner need to use IDesignerHook to avoid a requirement of designide at runtime. In runtime code, IDesignerHook should suffice. Design-time code can use IDesigner, but should use something like

```
var
  RealDesigner: IDesigner;
...
SomeDesignerHook.QueryInterface(IDesigner,RealDesigner);
...
```

to get the real IDesigner interface from an instance of IDesignerHook. IDesignerHook only requires Classes and Forms to be available. IDesigner requires DesignIntf, which includes many other packages, some of which may not be redistributable.

Borland wishes to thank field tester Matt Palcic for bringing these changes to our attention.

Component editor changes

Compatibility issues

The class TComponentEditor has a different ancestry in Delphi 6. In Delphi 5, it descended from TInterfacedObject; it now descends from a new class, TBaseComponentEditor. Also, the class TComponentEditorClass is now a class of TBaseComponentEditor instead of TComponentEditor. These changes in hierarchy may require you to modify your older Delphi projects.

Borland wishes to thank field tester Clive Walden for bringing these changes to our attention.

TDesignWindow changes

Compatibility issues

There have been a number of changes related to the class TDesignWindow. It has been moved to the DesignWindows unit, and its FormClosed method has been replaced by DesignerClosed. In the past, one could obtain access to the form within FormClosed by using the AForm parameter. In DesignerClosed, it is now necessary to use Designer's Root property to access the form.

In FormClosed, one would create selection lists by calling TDesignerSelectionList.Create or TComponentList.Create. To create selection lists within DesignerClosed, it is necessary to use an IDesignerSelections interface. You can create one using the CreateSelectionList function.

Parameters for the SelectionClosed method are also different from what they were in Delphi 5.

Borland wishes to thank field tester Matt Palcic for bringing these changes to our attention.

VCL package changes

Compatibility issues

The contents of some of the VCL-related packages have been redistributed into other packages. If you made references to vcl50.dcp in your project, you will need to change those references to other units, such as vcl.dcp and rtl.dcp.

Borland wishes to thank field tester Clay Shannon for bringing this change to our attention.

OpenGL interface unit moved to rtl.dcp

Compatibility issues

The Borland OpenGL interface unit (opengl.dcu) was an independent unit in the Delphi 5 Lib folder. It has been incorporated into rtl.dcp in Delphi 6. This may cause some problems for Delphi 5 projects ported to Delphi 6.

Here is an example. In a Delphi 5 project, it was possible to make a project-specific override of the OpenGL unit by placing a unit with the same name somewhere in your project path. The same method used in Delphi 6 causes a name conflict in any component that uses rtl.dcp also, and a name change is now required.

Borland wishes to thank field tester John Williams for bringing this change to our attention.

Types moved from HTTPApp.pas to HTTPProd.pas

Compatibility issues

Several types in the HTTPApp unit have been moved to the HTTPProd unit. They are THTMLBgColor, THTMLAlign and THTMLVAlign. If your projects use any of these units, you should change your uses statements to refer to HTTPProd instead HTTPApp.

Search unit removed, SearchBuf moved and changed

Compatibility issues

The unit Search no longer exists in Delphi 6. The SearchBuf routine, which locates a substring within a text buffer, has been moved to the StrUtils unit and its parameters have changed. The final parameter is now a TStringSearchOptions object. If your project won't compile because the compiler can't find the Search unit, change your **uses** statement to include StrUtils instead of Search. You will also want to check your SearchBuf calls to ensure that your parameters match the new syntax.

Link not found

The topic you requested is either not available or not linked to this Help system. This can occur if you launched this Help file from a system on which Delphi has not yet been installed, or if the subject matter you are requesting is not available in your edition of Delphi.

The topic you requested is now loading. If it does not appear within a few seconds, the topic is either not available or not linked to this Help system. This can occur if you launched this Help file from a system on which Delphi has not yet been installed, or if the subject matter you are requesting is not available in your edition of Delphi.