##$$KKTest Container Help

Use Test Container to test your ActiveX control by changing its properties, invoking its methods, and firing its events.

Test Container can display logs of events and data-binding notifications. It also provides facilities to test a control's persistence by letting you save properties to a property bag, to a stream, or to a storage.

You can also use VBScript to automate your use of Test Container.

You can modify the features that the container implements. You can select which optional features the container implements. For examples, you can let the container support windowless controls. This ensures that your control will work in a variety of containers.

#

_ASUG_Contents
$$ Contents for Test Container Help
KK test container help contents; contents
help

#$$KGetting Control Help in the Test Container

In each dialog box, information is available for each control. There are three ways to get help on a control:

- Click the question mark box in the upper right-hand corner of the dialog box and with the help-activated pointer, select a control in the dialog box.
- Press the TAB key until the desired control is highlighted and then press F1.
- Right click on a control and the What's This button will appear. Click the What's This button to get the control help.

#

$ Getting Help in the Test Container
K help
_ASUG_Inserting_an_ActiveX_Control

#$$KInserting an ActiveX Control

Before testing your control, you must build and register it. You can manually register a control with the **Register Control** command on the **Tools** menu.

To insert a control into Test Container

1. Choose **Insert New Control** from the **Edit** menu.
2. Select a control.

If you see multiple entries with the same name, your registry may need to be cleaned.

#

#$$KInvoking Properties and Object Verbs

The Test Container's **Edit** menu has a **Properties...** option. If you select this option and if a control implements the **ISpecifyPropertyPages** interface, the container will display the controls property sheet and the included property pages.

If a control does not support the **ISpecifyPropertyPages** interface but does support aggregation, selecting the **Properties...** option from the **Edit** menu will display extended properties of the control in the Test Container.

Below the **Properties...** option in the **Edit** menu will be any one menu option for each object (OLE) verb that a control may support.

#

$ Invoking Properties and Object Verbs
K object verbs;verbs, object;property sheet of the control
_ASUG_Setting_the_level

#$$KSetting the Level of Debugging

You can set the level of output to the debugger to none, normal, or verbose.

To set the level of output to the debugger

1. From the **Options** menu, choose **Trace Level**.
2. Choose either **None**, **Normal**, or **Verbose**.

#

$$ KK Choosing Design or User Mode

You can use Test Container as a design tool or as a container similar to that in which users will view your control and/or form.

To choose Design or User mode, choose **Design Mode** from the **Options**.

When the menu command is selected, Test Container is in design mode. When it is cleared, it is in user mode.

Note This menu command serves the same purpose as selecting the UserMode ambient property.

#

$$ Choosing Design or User Mode
KK design mode, choosing;user mode, choosing
_ASUG_Choose_Which_Events

Use the Control Logging Options dialog box to specify the events, property changes, and property edit requests for which you want logging to occur.

To log events

1. From the **Control** menu, choose **Logging**.
2. Specify the events, property changes, and property edit requests for which you want logging to occur.

Tip You can use the **Freeze Events** command on the **Options** menu to toggle whether the container will ignore or accept events from the control. This command is the equivalent of **IOleControl::FreezeEvents**.

#

$$ KK Choosing a Location for Logging Output

Use the **Logging Options** dialog box to select a method of logging the events fired by your control.

To log events

1. From the **Options** menu, choose **Logging**.
2. Choose a location to which events will be logged.

#

$$ KK Invoking an Object’s Methods

Use the **Invoke Methods** dialog box to test a method’s behavior to different parameter values and types.

To call the control’s methods

1. From the **Control** menu, choose **Invoke Methods**.
2. Select the method’s name from the **Method Name** box.
3. Edit controls appear for methods with parameters. Enter the appropriate values.
4. Click **Invoke**.

#

Ambient properties are named characteristics of the container that apply to all controls in the container unless otherwise specified. Examples of ambient properties are default colors and fonts.

To change ambient properties

1. From the **Container** menu, choose **Ambient Properties**.
2. Choose a property name.
3. When **Enabled** is checked, you can specify appropriate values for the property value and/or property type.
4. Choose **Set Value**. If the **Property Type** is VT_COLOR or VT_FONT, select a color or font.

To add a non-standard property

1. From the **Container** menu, choose **Ambient Properties**.
2. Click the **New Property...** button.
3. Enter a property name and DISPID.
4. Click **OK**.

#

#$$KTesting Persistence

You can use a property bag, a stream, or storage to test the persistence of your control.

Property Bag

To save a single control using a property bag

1. Select a control.
2. From the **Control** menu, choose **Save to Property Bag**.
3. Click **OK**.

Stream

To save a single control using a stream

1. Select a control.
2. From the **Control** menu, choose **Save to Stream**.
3. Provide a filename using the **File name** and **Save as type** fields.

To insert a control from a stream

1. From the **Edit** menu, choose **Insert Control From Stream**.
2. Select the stream to open using the **Open** dialog box.

A new control is created and initialized from the previously saved version of the control.

Storage

To save a single control using storage

1. Select a control.
2. From the **Control** menu, choose **Save to Storage**.
3. Provide a filename using the **File name** and **Save as type** fields.

To insert a single control from storage

1. From the **Edit** menu, choose **Insert Control From Storage**.
2. Select the storage to open using the **Open** dialog box.

A new control is created and initialized from the previously saved version of the control.

#

$$ Testing Persistence
KK persistence, testing;property bag, using;stream, using;substorage, using
_ASUG_Saving_and_Restoring

#$$KSaving and Restoring a Session

You can save the Test Container environment for later use.

To save the session

When you have inserted more than one control into the test container and need to save their states, including their arrangement in the container:

1. From the **File** menu, choose **Save Session** as or **Save Session**.
2. Use the **File** name and **Save as** type fields to provide a filename.

To load the form from a stream

1. From the **File** menu, choose **New Session**.
2. Choose **Open Session**.
3. Select the saved session to restore using the **Open** dialog box.

#

$$ Saving and Loading a Form
KK form, saving and loading;saving a form;loading a form;stream, using; substorage, using
_ASUG_Using_Command-Line

$$ KK Using Command-Line Options to Customize Startup Mode

You can use a variety of command-line arguments to specify the startup mode for the ActiveX Control Test Container.

To provide arguments to the Test Container

1. From the **Tools** menu in Visual C++, choose **Customize**.
2. Select the **Tools** tab.
3. Choose **ActiveX Control Test Container**.
4. Enter one or more of the following options in the **Arguments** box.

ActiveX Control Test Container Options

-Ln	No logging
-Lo	Log to output window
-Ld	Log to debugger
-Lf *filename*	Log to file *filename*
-OQ+	Use quick activation
-OQ-	Do not use quick activation
-OT+	Use two-pass drawing
-OT-	Do not use two-pass drawing
-OI+	Support inactive controls
-OI-	Do not support inactive controls
-OW+	Allow windowless activation
-OW-	Disallow windowless activation
-U+	User mode
-U-	Design mode

#

#KK$$Automating the Test Container

With Automation, you perform tasks programmatically by writing VBScript macros or Developer Studio add-ins. VBScript macros are procedures you write in the VBScript language and add-ins are in-process COM components (DLLs) you write in Visual C++ or Visual Basic.

You can write a VBScript macro to use the Test Container in a non-interactive mode.

See the Visual C++ documentation for more information on Automation and VBScript.

Object Model

Extended Properties and Methods

Event Handling

Macros

#

K VBScript
$ Using Automation and the Test Container
# poot

#$Object Model

Test Container exposes its functionality to script programmers through a single object, called **TCForm**, in the script's global namespace. **TCForm** implements the following properties and methods:

- PrimarySelection
- InsertControl
- FindControl
- Log

KPrimarySelection

This property is the Automation interface of the primary selected control on the form. If exactly one control is selected, it is the primary selection. If two or more controls are selected, or if no controls are selected, the primary selection is Nothing. Setting this property selects the specified control and deselects all others. Setting this property to Nothing deselects all controls.

KInsertControl

This method inserts a new control into the form. The syntax is **TCForm.InsertControl** *ProgID*, *Name*.

ProgID

is a string representing the *ProgID* of the control to insert.

Name

is a string representing the value of the *Name* property for the new control. Test Container will expose the control to the script as a global variable with this name.

The **InsertControl** method returns the Automation interface of the new control.

KFindControl

The **FindControl** method gets the Automation interface of the control with the given name. The syntax is **TCForm.FindControl** *Name*.

Name

is the name of the control for which to retrieve the Automation interface.

The **FindControl** method returns the Automation interface of the control with the given name.

$ Object Model
K PrimarySelection, Automation property; Automation properties, PrimarySelection
K InsertControl, Automation property; Automation properties, InsertControl
K FindControl, Automation property; Automation properties, FindControl

KLog

The **Log** method writes a text message to the Test Container log. By default, all log output goes to the output window, but it can also be redirected to the debugger or to a file using the **Logging** menu option in the **Options** menu. The syntax is **TCForm.Log** *message*.

message

is the message to be sent to the log.

$

$#Extended Properties and Methods

Whenever a control supports aggregation, Test Container creates an extended control, which adds additional properties and methods to the control that are specific to Test Container. These additional properties and methods are accessed exactly as if they were implemented by the control itself.

There are five additional properties and methods:

- Name
- Activate
- Deactivate
- UIActivate
- UIDeactivate

KName

This property is the user-assigned name of the control and is used by scripts to refer to the control. This name should be unique with respect to any other controls on the form. If a control is created by the **TCForm.InsertControl** method, the initial value of the **Name** property is the *Name* parameter supplied in that method. If the control is inserted manually, it is assigned a default name based on the type of the control.

control.**Name** = "MyControl"

KActivate

The **Activate** method in-place activates the control. This is the same as selecting the control and choosing **Activate** from the **Control** menu.

control.**Activate**

KDeactivate

The **Deactivate** method in-place deactivates the control. This is the same as selecting the control and choosing **Deactivate** from the **Control** menu.

control.**Deactivate**

KUIActivate

The **UIActivate** method UI activates the control. This is the same as selecting the control and choosing **UI Activate** from the **Control** menu.

control.**UIActivate**

poot2
K Name, Automation extended properties; Automation extended properties, Name
K Activate, Automation extended properties; Automation extended properties, Activate
K Deactivate, Automation extended properties; Automation extended properties, Deactivate
K UIActivate, Automation extended properties; Automation extended properties, UIActivate

KUIDeactivate

The **UIDeactivate** method UI deactivates the control. This is the same as selecting the control and choosing **UI Deactivate** from the **Control** menu.

control.**UIDeactivate**

#

K UIDectivate, Automation extended properties; Automation extended properties, UIDectivate
poot3

#$$KEvent Handling

Test Container Automation scripts can contain functions that handle events generated by controls on the form.

Any VBScript Sub whose name is of the form *controlname_eventname* will automatically handle the *eventname* event of *controlname* control. This automatic association is created when the script is loaded. This means that any controls for which the script has event handlers must already exist on the form with the proper name when the script is loaded.

The easiest way to do this is to place all of the desired controls on the form, set each control's name using the Extended page on its property sheet, and then load the script that contains the event handlers. To avoid repeating these steps, you can then save the session to a file. Loading a session loads all of the controls on the form, and then loads all of the scripts that were saved in that session.

$

$ Event Handling
K event handling in Automation; Automation, event handling
$ Macros

$^{\#}\#^{K}$KMacros

In addition to handling events generated by controls, a Test Container script can contain subroutines that can be invoked directly by the user. These are referred to as macros. Any VBScript Sub with no parameters, and which is not declared as Private, can be invoked from Test Container using the **Macro** option on the **Tools** menu.

#

poot4
K Automation macros; macros, Automation
samples

#$K$$Samples

The TstCon MFC sample includes one or more subdirectories containing macros that show how to automate the Test Container. The TstCon sample gives you the code for the Test Container.

You can get access to the TstCon sample either by installing its files from the TstCon sample abstract or by going directly the MSDN CD that contains the Visual C++ samples.

#

K samples;automation samples
$ Samples
HIDC_PROPNAME

#Lists and allows you to select an ambient property of the container.
#

HIDC_DISPID

#Displays the Dispatch ID of the selected ambient property.
#

HIDC_ENABLED

#Specifies whether the controls on the page will see the ambient property.
#

HIDC_CHOOSECOLOR_AMBIENT

#Allows you to change the color associated with an ambient property.#

HIDC_CHOOSEFONT_AMBIENT

#Allows you to change the font associated with an ambient property.
#

HIDC_STATIC_PROPERTYVALUE

#Provides a space for you to type a property value for properties other than Color or Font.
#

HIDC_PROPVALUE

#Provides a space for you to type a property value for properties other than Color or Font.
#

HIDC_PROPTYPE

#Lists the current property type or enables you to select a new property type.
#

HIDC_SETVALUE_AMBIENT

#Sets the selected ambient property to the values shown in the Property Value box and the Property Type list.
#

HIDC_NEWPROPERTY

#Allows you to add a new ambient property.
#

HIDC_CATEGORIES

#Lists all available component categories on your machine and allows you to select one or more categories.#

HIDC_SELECTALL

#Selects all available component categories on your machine.
#

HIDC_OBJECTS

#Displays all of the controls currently in the container and provides information on their state: activated, windowless, etc.
#

HIDC_ALLOWWINDOWLESS

#Specifies that the controls will be windowless.
#

HIDC_TWOPASSDRAWING

#When checked, enables two-pass (flicker-free) drawing.
#

HIDC_IGNOREACTIVATEWHENVISIBLE

#Honors the IGNOREACTIVATEWHENVISIBLE flag.
#

HIDC_USEIPOINTERINACTIVE

#Specifies that the container will delay activating controls until necessary by using the IPointerInactive interface.
#

HIDC_USEQUICKACTIVATION

#Specifies that the container will implement the IQuickActivate interface. See the OCX96 specification for more information.
#

HIDC_IOLEINPLACESITEEX

#Specifies that the container will implement the IOleInPlaceSiteEx interface.
#

HIDC_IOLEINPLACESITEWINDOWLESS

#Specifies that the container will implement the IOleInPlaceSiteWindowless interface.
#

HIDC_IADVISESINKEX

#Specifies that the container will implement the IAdviseSinkEx interface. For more information, read about flicker-free drawing and the OCX96 specification.
#

HIDC_SBINDHOST

#Specifies that the container will implement SBindHost, requesting an object that allows you to bind to a given moniker. SBindHost usually implements the IBindHost interface.
#

H

#Specifies which events the container will log.
#

HIDC_PROPERTIES_CHANGES

#Specifies which property changes the container will log.
#

HIDC_PROPERTIES_EDITREQUESTS

#Lists properties that generate edit requests and allows you to select which edit requests the container will log.
#

HIDC_ALWAYS

#Specifies that the container will always allow this property edit.
#

HIDC_NEVER

#Specifies that the container will never allow the property edit.
#

HIDC_PROMPT

#Specifies that the container will prompt for a response to the property edit request.
#

HIDC_SELECTALL_EDITREQUESTS

#Selects all the properties in the EditRequest Properties list box.
#

HIDC_CONTROLS

#Lists the registered controls that you can insert into the test container.
#

HIDC_IMPLEMENTEDCATEGORIES

#Allows you to specify what categories of objects can be inserted into the test container.#

HIDC_REQUIREDCATEGORIES

#Allows you to specify which component categories the container will support.
#

HIDC_IGNOREREQUIREDCATEGORIES

#When selected, displays all controls regardless of whether they require specific container functionality.
#

HIDC_LOGTONULL

#Specifies no recording or display of control events, property changes, property edits, or logging output from scripts.
#

HIDC_LOGTOOUTPUT

#Specifies that logging output will be sent to the Output window.
#

HIDC_LOGTODEBUG

#Specifies that logging output will be sent to the Debugger window.
#

HIDC_LOGTOFILE

#Specifies that logging output will be sent to a specified file.
#

HIDC_FILENAME

#Displays or lets you specify the file that will receive logging data. You must exit test container to complete writing data to the file.#

HIDC_FILENAME_BROWSE

#Enables you to browse your hard disk directory structure for a file in which to store logging data.
#

HIDC_MACROS

#Shows the list of macros available in the current session. #

HIDC_OPENFILE

#Lets you load a file containing macros.
#

HIDC_METHODNAME

#Allows you to select a method or property to invoke.
#

HIDC_PARAMS

#Lists the parameters required by the selected method.
#

HIDC_PARAMVALUE

Enables you to select a parameter value for the currently selected parameter.
#

HIDC_STATIC_PARAMETERVALUE

Enables you to select a parameter value for the currently selected parameter.
#

HIDC_PARAMTYPE

#Lists the current type of parameter or enables you to select a new parameter type.
#

HIDC_SETVALUE

#Sets the selected parameter to the values shown in the Parameter Value box and the Parameter Type list.
#

HIDC_RETURNVALUE

#Displays the value returned by the most recently invoked method.
#

HIDC_EXCEPTIONSOURCE

#Displays the file that caused an exception in the most recently invoked method.
#

HIDC_EXCEPTIONDESC

#Displays a description of the exception.
#

HIDC_EXCEPTIONHELP

#Starts the Help file for the control's exception.
#

HIDC_INVOKE

#Invokes the selected method with the current parameters.
#

HIDC_CHOOSEFONT

#Allows you to change the font value of the parameter.
#

HIDC_CHOOSECOLOR

#Allows you to change the color value of the parameter. #

HIDC_PROPNAME_NEW

#Allows you to type a name for the new ambient property.
#

HIDC_DISPID_NEW

#Allows you to type a Dispatch ID for the new ambient property.
#

HIDC_OUTPUT

#Displays events and other logging information when you select “Log to output window” in the Logging Options dialog box (Click the Options menu, then choose Logging).
#

HIDC_CLEAROUTPUTWINDOW

#Clears event and other logging information from the Output window. (Button is gone)#

HIDC_PROPERTIES

#Creates a property bag and tells the control to persist its state into the property bag and then displays the contents of the property bag.
#

HIDC_REGISTEREDCONTROLS

#Lists all controls registered on your machine.
#

HIDC_UNREGISTER

#Unregisters a control from your machine by running DLLUnregisterServer and removing the control's class ID information from the registry.
#

HIDC_REGISTER

#Registers a control on your machine by allowing you to search your machine and select a file with a dll, .ocx or .ax file extension and then run DLLRegisterServer to put the control’s class ID information in your machine’s registry.
#

HIDC_REREGISTER

#Allows you to select and reregister one of the displayed controls.
#

HIDC_TABORDER

#Displays and enables you to modify the tab order of the controls in the test container; just select and drag a control name. The first control in the list is first in the tab order.
#

HIDD_TRACELEVEL

#