OLE 2.0 Object Viewer Help Index

Help File for Version 1.0

Written By Charlie Kindel
Microsoft Vertical Developer Relations.

What Is Ole2View?

Overview

Commands

Ole2View was developed using Microsoft Visual C++ and MFC 2.0.

What Is Ole2View?

The OLE 2.0 Object Viewer is a tool designed to help implementors of OLE 2.0 enabled applications better understand what is happening in their systems. It is also a powerful testing tool that lets you verify that your objects and interfaces are behaving exactly as you planned.

Primarily Ole2View has been designed to help developers answer the following questions:

- What OLE 2.0 objects are installed in my system? (The Object List Pane)
- What interfaces does a given object support? (The Interface List Pane)
- Is object x an inproc server or handler, or is it local? What are it's registration database entries? (The Object Information Pane)
- Does this object support this interface correctly? (Interface Viewer)
- What kind of information can i get from that interface?
- What does IDataObject report for clipboard formats?
- What does IDispatch support?
- What version are my OLE2 dlls? (Show OLE 2.0 Version)
- When an OLE2 app runs, where is it finding the OLE2 dlls? (Show OLE 2.0 Version)
- Does a given Type Library export functions, subroutines, constants, and variables correctly? If so, what are their names and types? (The Type Library Viewer)

However, because Ole2View is extensible (see Ole2View Interface Viewer DLLs), it is capable of answering many more of your OLE 2.0 questions!

How does it work?

In order to list all objects on your system, and to display the interfaces those objects support, Ole2View utilizies the following features of OLE 2:

- All OLE 2 objects have entries in the registration database.
- The registration data base contains a list of standard interfaces.

  How Ole2View Uses The Registration Database

- OLE 2 objects support a IClassFactory object (Class Object), a pointer to which will be returned by the CoGetClassObject() API (Class Object interface pointer).

  How Ole2View uses the Class Object

- IClassFactory::CreateInstance() will return an interface pointer to a new instance of an object (Instance interface pointer).

How Ole2View Uses The Registration Database

Central to the architecture of OLE 2 is the Windows registration database. The following briefly describes how OLE 2 and OLE 2 objects use the registration database, and how Ole2View uses the information stored there. For more information on the registration database see the Windows SDK documentation as well as Appendix A of the OLE 2 Programmers' Reference ("Registering Applications").

HKEY_CLASSES_ROOT\CLSID

Off of the root of HKEY_CLASSES_ROOT is a section with a key that equals "CLSID". This key contains entries for each OLE object installed on your system. Each entry (sub-key) has the form:

{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} = yyyyyy

where {xx...xx} is the unique CLSID of an object, and yyyyyy is the name of the object as it would appear in the "Object.Insert..." dialog box.

Each CLSID entry has a number of sub-keys that describe the object. For example, the location of the object's executable image and default icon. Ole2View uses the following sub-keys:

HKEY_CLASSES_ROOT\CLSID\<clsid>\DefaultIcon

Used to determine the object's icon.*

HKEY_CLASSES_ROOT\CLSID\<clsid>\LocalServer

Used to determine the object's icon.*

HKEY_CLASSES_ROOT\CLSID\<clsid>\InprocServer

Used to determine the object's icon.*

HKEY_CLASSES_ROOT\CLSID\<clsid>\InprocHandler

Used to determine the object's icon.*

HKEY_CLASSES_ROOT\CLSID\<clsid>\ProgID

Contains the Programmatic Identifier for the object, and used to reference the OLE 1.0 compatiblity information stored under the ProgID key (see below).

HKEY_CLASSES_ROOT\CLSID\<clsid>\Ole1Class

Identifies the object as an OLE 1.0 object. Ole2View uses the existence of this key to determine whether the listbox icon for the object should have a little "1.0" in it.

HKEY_CLASSES_ROOT\CLSID\<clsid>\Control

Identifies the object as an OLE Custom Control.

HKEY_CLASSES_ROOT\<progid>\Insertable

When present, the object is an embeddable OLE 2 object. Ole2View uses this entry to determine if the object should have an "Insertable" icon in the Object List Pane.

*To find the object's icon Ole2View tries the DefaultIcon, LocalServer, InprocServer, InprocHandler keys in the order given. If it cannot find the icon using these keys, it tries the \<ProgID>\Protocol\StdFileEditing\Server entry, and if that fails it display's a default "OLE 2" icon.

Also, if the {CLSID} entry has an InProcServer key whose value is "ole2prox.dll", Ole2View identifies the object as being a "ProxyStub" object. These objects will not appear in the Object List Pane unless the Show OLE 2 "PS*" Objects menu item is checked.

HKEY_CLASSES_ROOT\<ProgID> = MainUserTypeName

Ole2View displays the contents of these keys, but only *uses* the Protocol\StdFileEditing\Server sub-key (to determine the object's icon; see above).

HKEY_CLASSES_ROOT\Interfaces

OLE 2 provides one way for determining if an object supports a particular interface: The IUnknown::QueryInterface() function. Because QueryInterface() requires that a interface identifier (IID) be passed, there is no way to 'enumerate' the interfaces supported by an object.

However, the registration database contains a key with the name "Interfaces" that contains sub-keys for each standard OLE 2 interface. These subkeys have the following form:

{xxxxxxxx-xxxx-xxxx-xxxx-xxxxxxxxxxxx} = yyyyyy

Where "{xxx....xxx}" is the Interface ID of the interface and "yyyyyy" is the textual name (e.g. "IDataObject"). These keys are referred to as {IID} keys.

Thus, in order to determine what interfaces an object supports, Ole2View simply enumerates the \Interfaces key in the registry (using RegEnumKey), and calls QueryInterface() for each one.

How Ole2View uses the Class Object

When you double click on an object in the Object List Pane, Ole2View exectutes code similar to the following:

```
// Get the Class Object
hr = CoGetClassObject( m_clsidCur, m_dwClsCtx, NULL,
                       IID_IClassFactory,
                       (LPVOID FAR *)m_pIClassFactory ) ;

if (SUCCEEDED(hr))
{
    // Create an instance of the object (Instance) and store
    // it in m_pIUnknown
    //
    hr = m_pIClassFactory->CreateInstance( NULL, IID_IUnknown,
                                  (LPVOID FAR *)&m_pIUnknown ) ;
}

for (each entry in the \Interfaces section of the registry)
{
    hr = m_pIUnknown->QueryInterface( iid, (LPVOID FAR*)&pIUnk ) ;
    if (SUCCEEDED( hr ))
    {
        // Add this entry to the listbox and identify it as an interface
        // that was retrieved via QueryInterface on the Instance object
        ...
        pIUnk->Release() ;
    }

    hr = m_pIClassFactory->QueryInterface( iid, (LPVOID FAR*)&pIUnk ) ;
    if (SUCCEEDED( hr ))
    {
        // Add this entry to the listbox and identify it as an interface
        // that was retrieved via QueryInterface on the Class Object
        ...
        pIUnk->Release() ;
    }
}
```

where m_clsidCur is the CLSID of the currently selected object in the object list pane, and m_dwClsCtx is determined by the CLSCTX menu/toolbar items.

File menu commands

The File menu offers the following commands:

Bind To File	Create an instance of an object by binding to a persistant instance of that object (e.g. a file) by using a file moniker.
View Type Library	Allows you to view (browse) a typelibrary file.
Run the Registration Editor	Runs REGEDIT.EXE /V.
Show OLE 2.0 Version	Shows OLE 2.0 Version Information.
Exit	Exits Ole2View.

Bind To File (File Menu)

Use this command to create an instance of an object by binding to a persistant instance of that object (e.g. a file) by using a file moniker.

When you choose this command, a File Open dialog box appears; allowing you to specify a file to be bound to. Ole2View takes the filename you provide and calls the CreateFileMoniker API to create a file moniker for the file. It then calls the BindMoniker API to attempt to bind to the file moniker.

Ole2View can also accept a filename on the command line to accomplish the same thing.

View Type Library Command (File Menu)

Use this command to view a type library file (these files are generated by the MkTypeLib utility).

Output TypeLib to File

The Type Library Viewer has a "To File..." button. This button allows you to dump the contents of a TypeLibrary (or TypeInfo) to a text file. The file will contain all of the functions, subroutines, variables, and constants found in the type library in a Visual Basic like syntax. The entries in this file look like this:

Functions

```
' GetFileData
'    Retrieves the date stamp of a file
Declare Function GetFileDate (FileName As String) As String
```

Subroutines (Functions without return values)

```
' NewDocument
'    Creates a new document
Declare Sub NewDocument (Author As String, FileName As String, Revision As Integer)
```

Variables

```
' CurrentUser
'     The currently logged in username
Dim CurrentUser As String
```

Constants

```
' MAX_COLORS
'    The maximum number of colors supported.
Const MAX_COLORS As Integer = 256
```

The "To File..." feature is particularly useful when testing your product. It allows you to create a file that you can compare against your documentation (or to create your documentation!).

See the OLE 2.0 Automation documentation for more information on type libraries.

The code for the Type Library Viewer is actually contained within DEFO2V.DLL, and is the same code that is invoked when either the IDispatch Interface Viewer or the ITypeInfo Interface Viewers are used. See Interface Viewer for more information.

Show OLE 2.0 Version Command (File Menu)

Use this command to display the OLE 2 version numbers along with the paths where the OLE 2.0 DLLs were loaded from.

How to use this feature

This command is useful when you are not sure if you have multiple versions of the OLE DLLs on your system and you want to make sure the right versions are being loaded.

1. Make sure no OLE 2 applications are running (including Ole2View!).
2. Use the "OLE 2.0 Free Module" application included with the OLE 2.0 SDK to free any residual OLE 2.0 DLL instances.
3. Run the OLE 2.0 application you are curious about to load the OLE 2.0 DLLs. You may have to embed an object in order for the libraries to be instantiated.
4. Run Ole2View and choose the Show OLE 2.0 Version Command. The list of DLLs will show you where the DLLs were found.

Run the Registration Editor (File Menu)

Use this command to run the Registration Database Editor.

(Someday this command will be unneccessary because Ole2View will provide a facility for editing OLE 2.0 registry information...)

Exit command (File menu)

Use this command to end your Ole2View session. You can also use the Close command on the application Control menu. Ole2View prompts you to save documents with unsaved changes.

Shortcuts

Mouse: Double-click the application's Control menu button.

Keys: ALT+F4

Objects menu commands

The Objects menu offers the following commands:

Show Class IDs	Shows or hides class IDs in the Object List Pane.
Show OLE2 'PS*' Objects	Shows or hides Proxy Stub objects (internal OLE 2.0 objects). These objects typically have names that begin with "PS" and do not support any documented OLE 2.0 interfaces.

Class ID command (Objects menu)

Use this command to display or hide class IDs in the Object List Pane .

Sometimes it is useful to have a quick reference as to which class IDs are assigned to which objects in your system. Turning this option on provides a way to easily see this.

Show OLE 'PS*' Objects command (Objects Menu)

Shows or hides OLE 2.0 Proxy Stub objects that are identified as internal OLE 2.0 objects. These objects typically have names that begin with "PS" and do not support any documented OLE 2.0 interfaces (they are used by the OLE 2.0 Proxy manager...they are "ProxyStubs", hence the PS in the names).

Because these objects do not support any published interfaces, displaying them is of little value.

Ole2View identifies an object as an Proxy Stub object if it's InprocServer entry in the registration database has a value of "ole2prox.dll" or "ole2disp.dll".

Objects that are identified as internal to OLE 2.0 have a question mark icon associated with them.

Interfaces menu commands

The Interfaces offers commands that control how the Interface List Pane works:

Get Interfaces	Instantiates the current object and shows implemented interfaces.
Interface Viewer...	Displays the Interface Viewer for the currently selected interface if that interface is implemented.
Show All Interfaces	Turns on or off the display of all interfaces in the Interface List Pane
Use CLSCTX_INPROC_SERVER	If this item is selected, Ole2View will use the CLS_CTX_INPROC_SERVER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object
Use CLSCTX_INPROC_HANDLER	If this item is selected, Ole2View will use the CLS_CTX_INPROC_HANDLER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object.
Use CLSCTX_LOCAL_SERVER	If this item is selected, Ole2View will use the CLS_CTX_LOCAL_SERVER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object.

See Also Interface List View

Get Interfaces command (Interfaces Menu)

This command causes the Interface List Pane to instantiate the currently selected object and to display the interfaces that that object implements.

Interface Viewer... Command (Interfaces Menu)

Displays the Interface Viewer for the currently selected interface if that interface is implemented.

Show All Interfaces command (Interfaces Menu)

Use this command to display or hide Interfaces that are not supported by the current object in the interface list box.

While interfaces that are supported by an object are clearly identified in the Interface List Pane by their icon and **bold text**, it is sometimes helpful to turn off the display of the interfaces that are not implemented.

Show ClassFactory Object Interfaces Command (Interfaces Menu)

Use this command to show or not show interfaces that are implemented by the IClassFactory Object (Class Object).

See Interface List Pane for more information.

Use CLSCTX_INPROC_SERVER Command (Interfaces Menu)

If this item is selected, Ole2View will use the CLS_CTX_INPROC_SERVER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object.

Using this command causes Ole2View to unload the currently active object, releasing interface pointers, and then reload the object using the new CLSCTX flags.

Use CLSCTX_INPROC_HANDLER Command (Interfaces Menu)

If this item is selected, Ole2View will use the CLS_CTX_INPROC_HANDLER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object.

Using this command causes Ole2View to unload the currently active object, releasing interface pointers, and then reload the object using the new CLSCTX flags.

Use CLSCTX_LOCAL_SERVER Command (Interfaces Menu)

If this item is selected, Ole2View will use the CLS_CTX_LOCAL_SERVER flag when calling the OLE 2.0 function CoGetClassObject to retrieve the IClassFactory pointer for the object.

Using this command causes Ole2View to unload the currently active object, releasing interface pointers, and then reload the object using the new CLSCTX flags.

View menu commands

The View menu offers the following commands:

Class IDs	Shows or hides class IDs in the object list.
Refresh	Causes Ole2View to unload the current object, re-read the registration database, reinstantiate the current object, and redisplay it's supported interfaces.
Toolbar	Shows or hides the toolbar.
Status Bar	Shows or hides the status bar.

Refresh command (View Menu)

Causes Ole2View to unload the current object, re-read the registration database, reinstantiate the current object, and redisplay it's supported interfaces.

Toolbar command (View menu)

Use this command to display and hide the Toolbar, which includes buttons for some of the most common commands in Ole2View, such as File Open. A check mark appears next to the menu item when the Toolbar is displayed.

See Toolbar for help on using the toolbar.

Status Bar command (View menu)

Use this command to display and hide the Status Bar, which describes the action to be executed by the selected menu item or depressed toolbar button, and keyboard latch state. A check mark appears next to the menu item when the Status Bar is displayed.

See Status Bar for help on using the status bar.

Help menu commands

The Help menu offers the following commands, which provide you assistance with this application:

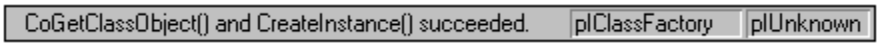
Index	Offers you an index to topics on which you can get help.
Using Help	Provides general instructions on using help.
About	Displays the version number of this application.

Toolbar

The toolbar is displayed across the top of the application window, below the menu bar. The toolbar provides quick mouse access to many tools used in Ole2View,

To hide or display the Toolbar, choose Toolbar from the View menu (ALT, V, T).

Status Bar



The status bar is displayed at the bottom of the Ole2View window. To display or hide the status bar, use the Status Bar command in the View menu.

The status bar describes actions of menu items as you use the arrow keys to navigate through menus. This area similarly shows messages that describe the actions of toolbar buttons as you depress them, before releasing them. If after viewing the description of the toolbar button command you wish not to execute the command, then release the mouse button while the pointer is off the toolbar button.

Index command (Help menu)

Use this command to display the opening screen of Help. From the opening screen, you can jump to step-by-step instructions for using Ole2View and various types of reference information.

Once you open Help, you can click the Contents button whenever you want to return to the opening screen.

Using Help command (Help menu)

Use this command for instructions about using Help.

About command (Help menu)

Use this command to display the copyright notice and version number of your copy of Ole2View.

Context Help command

Use the Context Help command to obtain help on some portion of Ole2View. When you choose the Toolbar's Context Help button, the mouse pointer will change to an arrow and question mark. Then click somewhere in the Ole2View window, such as another Toolbar button. The Help topic will be shown for the item you clicked.

Shortcut

Keys: SHIFT+F1

Title Bar

The title bar is located along the top of a window. It contains the name of the application and document.

To move the window, drag the title bar. Note: You can also move dialog boxes by dragging their title bars.

A title bar may contain the following elements:

- Application Control-menu button
- Document Control-menu button
- Maximize button
- Minimize button
- Name of the application
- Name of the document
- Restore button

Scroll bars

Displayed at the right and bottom edges of the document window. The scroll boxes inside the scroll bars indicate your vertical and horizontal location in the document. You can use the mouse to scroll to other parts of the document.

Size command (System menu)

Use this command to display a four-headed arrow so you can size the active window with the arrow keys.

After the pointer changes to the four-headed arrow:

1. Press one of the DIRECTION keys (left, right, up, or down arrow key) to move the pointer to the border you want to move.
2. Press a DIRECTION key to move the border.
3. Press ENTER when the window is the size you want.

Note: This command is unavailable if you maximize the window.

Shortcut

Mouse: Drag the size bars at the corners or edges of the window.

Move command (Control menu)

Use this command to display a four-headed arrow so you can move the active window or dialog box with the arrow keys.

Note: This command is unavailable if you maximize the window.

Shortcut

Keys: CTRL+F7

Minimize command (application Control menu)

Use this command to reduce the Ole2View window to an icon.

Shortcut

Mouse: Click the minimize icon  on the title bar.

Keys: ALT+F9

Maximize command (System menu)

Use this command to enlarge the active window to fill the available space.

Shortcut

Mouse: Click the maximize icon  on the title bar; or double-click the title bar.

Keys: CTRL+F10 enlarges a document window.

Close command (Control menus)

Use this command to close the active window or dialog box.

Double-clicking a Control-menu box is the same as choosing the Close command.

Note: If you have multiple windows open for a single document, the Close command on the document Control menu closes only one window at a time. You can close all windows at once with the Close command on the File menu.

Shortcuts

Keys: ALT+F4 closes the OLE 2.0 Object Viewer window or dialog box

Restore command (Control menu)

Use this command to return the active window to its size and position before you chose the Maximize or Minimize command.

Switch to command (application Control menu)

Use this command to display a list of all open applications. Use this "Task List" to switch to or close an application on the list.

Shortcut

Keys: CTRL+ESC

Dialog Box Options

When you choose the Switch To command, you will be presented with a dialog box with the following options:

Task List

Select the application you want to switch to or close.

Switch To

Makes the selected application active.

End Task

Closes the selected application.

Cancel

Closes the Task List box.

Cascade

Arranges open applications so they overlap and you can see each title bar. This option does not affect applications reduced to icons.

Tile

Arranges open applications into windows that do not overlap. This option does not affect applications reduced to icons.

Arrange Icons

Arranges the icons of all minimized applications across the bottom of the screen.

Next Pane

F6 of TAB Switches to the next pane.

Prev Pane

Shift-F6 or Shift-TAB Switches to the previous pane.

No Help Available

No help is available for this area of the window.

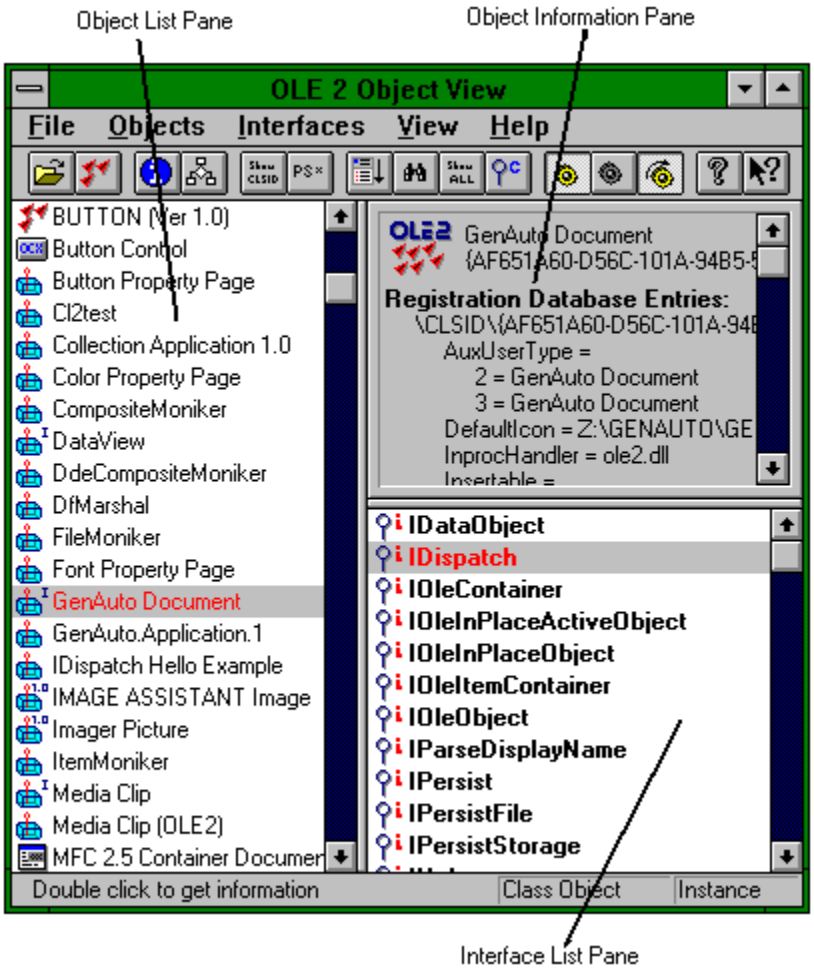
No Help Available

No help is available for this message box.

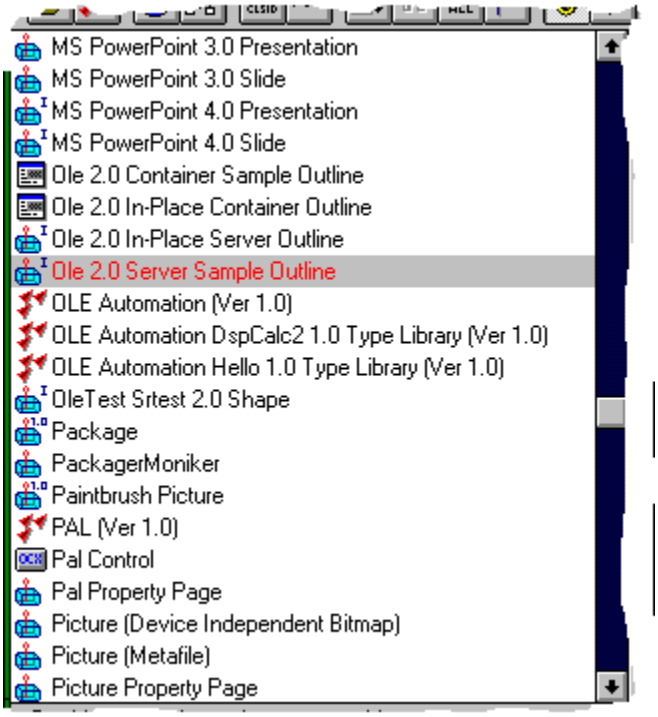
The Main Ole2View Window

The main Ole2View window is made up of the following parts:

The Menu
The Toolbar
The Object List Pane
The Object Information Pane
The Interface List Pane
The Status Bar



The Object List Pane



Ole2View's main window is divided into three panes. The pane on the left is the *Object List Pane*.. The Object List Pane shows a list of all OLE objects installed in your system. This list is generated by enumerating the CLSID keys in the registration database.

Each entry in the list has an icon associated with it. These icons are described below:

This object is a generic OLE 2.0 object. It is not insertable in containers (i.e. it does not have the "Insertable" key in it's section in the registration database). This type of object will not show up in the list of objects a user can insert in a container.

Examples of this type of object might include applications that support OLE 2.0 Automation only (such as the IDispatch Calculator sample program), or data objects that support only the IDataObject interfaces for Uniform Data Transfer.

This object is an *Insertable* OLE 2.0 object. In otherwords, it can be inserted into a container using the Insert Object command of the container application. This type of object indicates that it is insertable by having the "Insertable" key in it's section in the registration database.

The standard OLE 2.0 "Insert Object" dialog box uses the "Insertable" key to determine whether an object should be listed or not.

This is an OLE 1.0 object.

An OLE 2.0 Automation Type Library. Double clicking on one of these will invoke the Type Library Viewer.

Double clicking on a Type Library in the Object List Pane has the same effect as using the View Type Library menu item and choosing the corresponding type library filename.

An OLE Custom Control (.OCX).

An OLE 2.0 Container Application.

Only those containers that support linking to embeded objects appear in the Object List Pane.

Objects that are identified as internal to OLE 2.0 have a question mark icon associated with them. Because these objects do not support any published interfaces, displaying them is of little value.

You can use the Show OLE 2 'PS*' Objects command to turn the display of these objects on and off.

Note that some of the objects listed in this list are not real OLE objects; at least they are not real in the sense that they do not support IClassFactory and calling CoGetClassObject() on their class IDs fails. These objects are used internally by OLE 2.0. Examples are all objects whose names begin with "PS", such as PSBindCtx etc...

How the Object List works

The Object List fills itself by enumerating the HKEY_CLASSES_ROOT\CLSID key of the registration database using the Windows SDK API RegEnumKey().

See How Ole2View Uses The Registration Database for more information on how objects are identified in the Registration Database.

Showing Object Information

When you select an object in the object list the Object Information Pane is notified and it displays all the information found in the registration database regarding the object.

When you double click on an object in the Object List Pane the Interface List Pane is notified. The Interface List Pane then *instantiates* that object and displays a list of all interfaces implemented by the object.

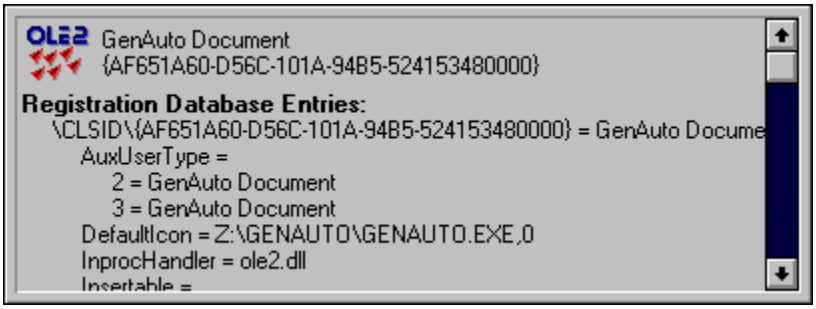
An object remains active (the pointer Ole2View keeps for the IClassFactory interface is valid) until you click on another object in the Object List or shut Ole2View down. You can tell that an object is active by the indicator in Ole2View's status bar. If the status bar reads, "Object Active" the currently selected object is active.

Showing Class IDs in the Object List

The "Show CLSIDs" command in the Objects menu, and on the toolbar, allows you to turn on and off the display of the Class IDs associated with each object in the Object List Pane.

The Object Information Pane

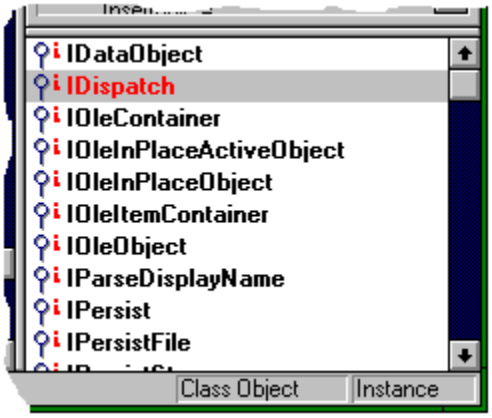
Ole2View's main window is divided into three panes. The pane on the top-left is the ***Object Information Pane***.



The Object Information Pane displays all the information about the currently selected OLE 2.0 object that can be ascertained by looking in the registration database. For more information on what is listed here see the Registering Object Applications section in the OLESDKV1.HLP help file (or the OLE 2.0 Programmers Reference, Chapter 3).

Currently the information listed is 'static'; that is, Ole2View will not let you edit it. Someday Ole2View may be enhanced to allow easy editing of all this information. For now you can run the Registration Database Editor (REGEDIT) from the File menu.

The Inteface List Pane



This window displays the list of interfaces supported by the currently selected object. Interfaces that are supported are displayed in **bold text**. Those that are not are either not displayed at all or are in normal text (use the View All Interfaces command to change this behavior).

Each interface listed in the Interface List Pane has an icon assoicated with it. The following table describes each icon:

	This interface is implemented by the Object's "Class Factory Object". That is, the interface pointer was returned by a call to IClassFactory::QueryInterface(). Often, when an object relies on the OLE 2.0 libraries for default implementation there will be many interfaces listed of this type. Note that the display of IClassFactory Object interfaces can be turned on and off using the Show ClassFactory Interfaces command.
	This icon indicates that the interface is implemented by the instance of the object you have selected. That is, the interface pointer was returned by a call to IUnknown::QueryInterface(), where the IUnknown pointer was retrieved by calling IClassFactory::CreateInstance(). Interfaces with this icon are the interfaces that are typically thought of as those 'implemented by your object'.
	This icon indicates that the interface is not implemented by either the object or the object's IClassFactory Object. You can turn the display of non-implemented interfaces by using the Show All Interfaces command.

How it works

The Interface List Pane figures out what interfaces an object implements by first *instantiating* the object, and then trying each known interface ID using QueryInterface. That is, Ole2View calls the OLE 2.0 function, CoGetClassObject() and retrieves a ponter to that objects IClassFactory interface, then uses pIClassFactory->CreateInstance() to get a pointer to the object's IUnknown interface (pInterface).

Once the Interface List has a pointer to the object's IUnknown, it enumerates the HKEY_CLASSES_ROOT\Interfaces key in the registration database, trying each interface identifier (IID) found there by calling pIUnknown->QueryInterface(). If a call to QueryInterface is successfull, the Interface List code knows that the object implements that interface. If QueryInterface fails, the object does not implement that interface.

The Interface List stores any interface pointers returned by QueryInterface with it's respective item in the list box. Thus, when you double click on a supported interface, the Interface Viewer has an interface pointer to work with.

NOTE: An object remains active (the pointer Ole2View keeps for the IClassFactory interface is valid) until you click on another object or shut Ole2View down. You can tell that an object is active by the indicator in Ole2View's status bar. If the status bar reads, "Object Active" the currently selected object

is active.

Making Ole2View See Other Interfaces

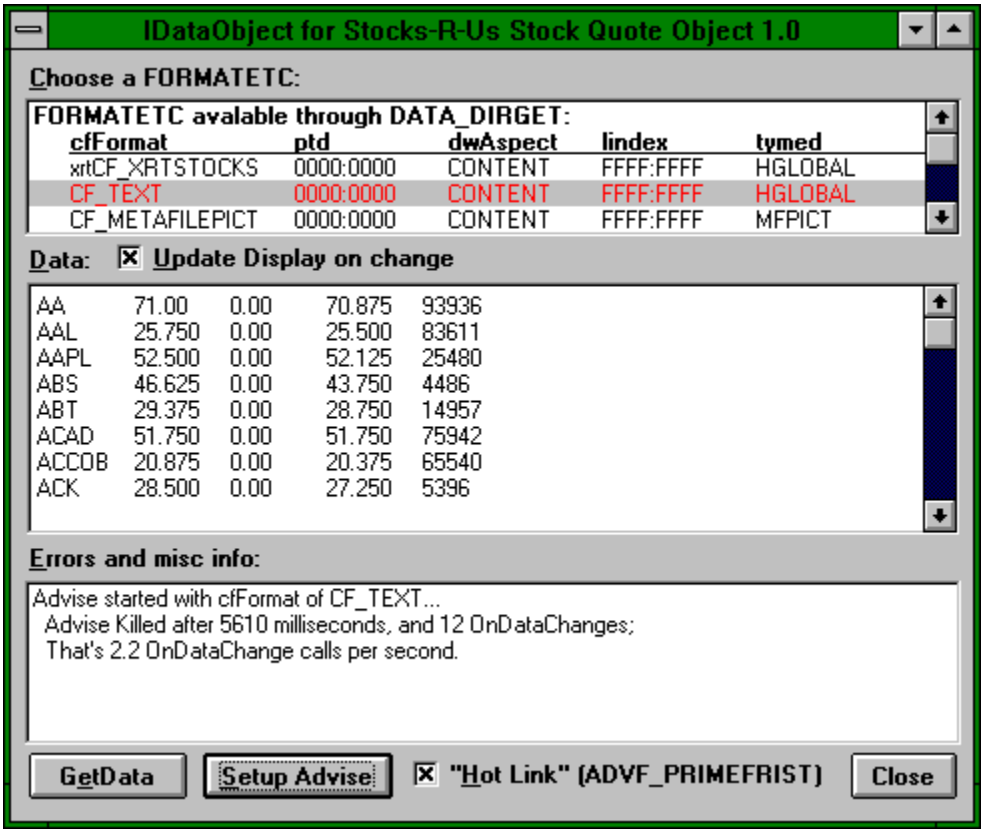
Since Ole2View uses the Interfaces section of the registration database to determine what interfaces an object supports, you can make it see newly defined interfaces by adding entries to this section.

Interface Viewer

When you double click on an interface in the Interface List Pane, Ole2View attempts to execute an Interface Viewer for that interface. Interface Viewers are functions exported by DLLs that take a pointer to an interface and do whatever is neccessary to display whatever information is available from that interface.

Ole2View includes a DLL (DEFO2V.DLL) with 2 of these Interface Viewers in it:

IDataObject Interface Viewer

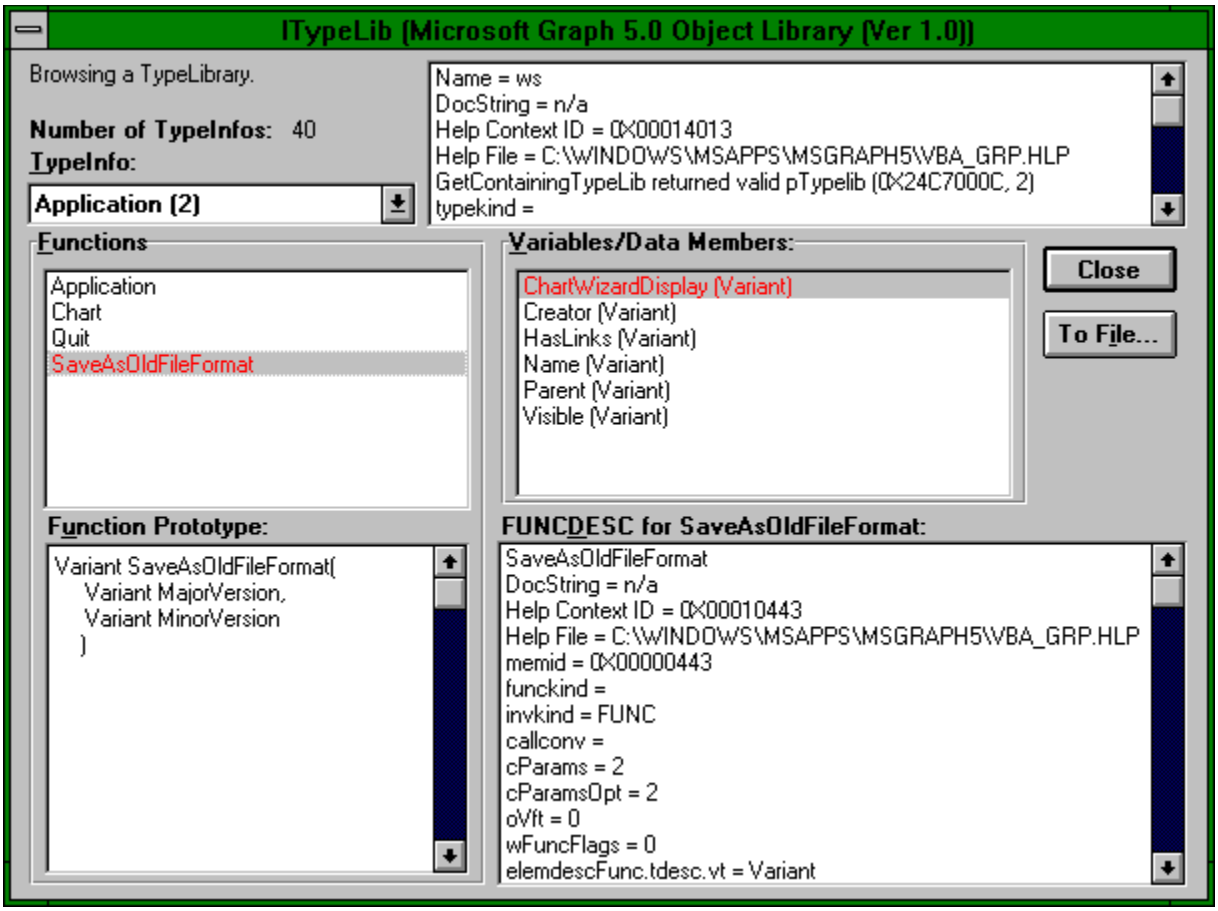


Displays the enumeration of FORMATETC structures that the object supports for both DATADIR_GET and DATADIR_SET.

Allows you to call members on the IDataObject pointer to test the interface. You can "GetData" and setup an advise link (using DAdvise). The clipboard format used for GetData and DAdvise are determined by the currently selected clipboard format in the FORMATETC listbox. The only clipboard format the the viewer currently understands is CF_TEXT.

You can easily add support for more clipboard formats by modifying the source code found in the DEFO2V sample directory.

IDispatch/ITypeInfo/TypeLib Interface Viewer



Displays the Type Information for the object. It is essentially a typeinfo browser. Note that the same code that is called in DEFO2V.DLL to view an IDispatch or ITypeInfo, is called when you choose the View Type Library command.

See also: Ole2View Interface Viewer DLLs

Ole2View Interface Viewer DLLs

Ole2View is extensible in that it can call functions in DLLs to display interface information. These DLLs are called Ole2View Interface Viewers.

Interface Viewers are called when you double click on an interface in the Interface List Pane .

Ole2View comes with one Inteface Viewer DLL (DEFO2V.DLL) which contains interface viewers for IDataObject, IDispatch, ITypeInfo, and ITypeLib. *The source code is included to provide an example of how to implement an Interface Viewer for Ole2View, and to provide sample code for both the IDataObject and IDispatch interfaces. See the DEFO2V sub-directory in the OLE2\SAMPLES directory .*

The "DisplayInterface" Function

An Ole2View Interface Viewer is a single function exported from any DLL prototyped as:

```
HRESULT WINAPI _export
DisplayInterface
(
    HWND            hwndParent, // Ole2View's main frame wnd
    LPUNKNOWN       lpunk,      // Pointer to valid interface
    LPIID           lpiid,      // pointer to interface id
    LPSTR           lpszName    // Interface name
) ;
```

Your DLL should export the DisplayInterface functions by name.

For example, DEFO2V.DLL includes DisplayInterface functions for both IDataObject and IDispatch (DisplayIDataObject and DisplayIDispatch respectively). They are prototyped within the DEFO2V source code as:

```
HRESULT WINAPI _export
DisplayIDataObject( HWND hwndParent, LPDATAOBJECT lpDO,
                    LPIID lpiid, LPSTR lpszName ) ;

HRESULT WINAPI _export
DisplayIDispatch( HWND hwndParent, LPDISPATCH lpDisp,
                  LPIID lpiid, LPSTR lpszName ) ;
```

and are included in the EXPORTS section of the DEFO2V.DEF file.

You can do whatevery you wish within a DisplayInterface function, as long as you follow standard Windows DLL programming rules. The information passed to the DisplayInterface functions should be enough to allow you to do just about anything.

The [Interface Viewers] Section Of OLE2VIEW.INI File

Ole2View looks in the [Interface Viewers] section of the OLE2VIEW.INI file to determine where to find installed interface viewers.

The format of an entry in the [Interface Viewers] section is:

<interface id>=<pathname of dll>, <displayinterface function name>

For example:

[Interface Viewers]

{0000010E-0000-0000-C000-000000000046}=defo2v.dll,DisplayIDataObject
{00020400-0000-0000-C000-000000000046}=defo2v.dll,DisplayIDispatch

If you wanted an "IOleContainer" interface viewer you would create a DLL named, say, "contain.DLL" and export the "DisplayIOleContainer" function from it. Then the follwing line in the OLE2VIEW.INI file would enable Ole2View to find it.

{0000011B-0000-0000-C000-000000000046}=contain.dll,DisplayIOleContainer

Ole2View knows internally about the interface viewers in DEFO2V.DLL. It is not necessary to have entries for them in OLE2VIEW.INI. However, you can override the default entries if you wish.