Related information

- *Windows User's Guide:* Chapter 4, "File Manager"; Chapter 9, "Write"; Chapter 10, "Paintbrush"; Chapter 13, "Integrating Your Windows Applications"

Chapter

11 Compound Documents in Windows 3.1

This chapter presents an overview of how Object Linking and Embedding (OLE) works when a Windows application is designed to take advantage of the OLE features of Microsoft Windows 3.1. This chapter also describes how to update and add to the system registration database using REGEDIT.EXE.

- *Windows Resource Kit:* Chapter 6, "Tips for Configuring Windows 3.1"
- *Glossary terms:* dynamic data exchange (DDE) and dynamic-link library (DLL)

Contents of this chapter

Flowchart 5.10 Problems with OLE

This chapter explains how OLE works in Windows 3.1 and how you can take advantage of it to create and use compound documents. The information here provides a broad overview. For details about how a specific Windows application implements OLE, see that application's documentation.

About Object Linking and Embedding

You can share data between Windows applications in several ways:

- You can use the Cut and Copy commands to place data on the Clipboard, and then use the Paste command to place the data in another kind of document or file. (You can also use Cut and Paste to share data between Windows applications and many non-Windows applications.)
- In many Windows applications, you can also take advantage of dynamic data exchange (DDE) to retrieve and update information dynamically. For example, you can use the Paste Link command to link data from one Microsoft Excel spreadsheet to another spreadsheet or to a Microsoft Word for Windows document. The data is updated dynamically in the second spreadsheet or Microsoft Word document whenever it is edited in the original spreadsheet.

W

- For Windows applications that use Object Linking and Embedding (OLE) protocol, you can place data from one application in another kind of document as an object or as an iconic package. Whenever you want to update or view the embedded data, you can double-click it to run the application in which the data was originally created.

About Compound Documents

Any Windows application that uses OLE can work with other OLE applications to produce documents that contain different kinds of data. You can view or modify data in a compound document without knowing what applications were used to create the original data. Whenever you double-click a part of the compound document to work with it, the application in which it was created

will start automatically.

A compound document can contain formatted text, graphics, and data from a spreadsheet or a database. It can also contain icons that run sound recordings or play multimedia devices. The more hardware you have to support creating and playing multimedia through sound boards, CD-ROM, and other media players, the more rich a compound document you can create. But even with a basic PC system, you can take advantage of OLE to create compound documents.

You may have already seen compound documents. For example, a Microsoft Excel spreadsheet with embedded graphics and detail reports is a compound document. The README files for Windows applications such as Microsoft Word for Windows sometimes contain icons in the table of contents that you click for navigation.

Compound documents created using Object Linking and Embedding in Windows is one of the first manifestations of Information At Your Fingertips™, the Microsoft vision for personal computing in the 1990s.

About Objects, Packages, Clients, and Servers

An *object* in a compound document is any encapsulated data that can be displayed and manipulated by a user. For example, a single cell, a range of cells, or an entire spreadsheet could be embedded as an object in a word processing text file. Any data can be an object if it was created in a Windows application that uses OLE.

A *package* is an embedded icon that contains an object, a file or part of a file, or a command.

The *client application* is any Windows application that can accept, display, and store objects. Usually a client application displays a distinctive border or some other visual cue so that you can identify an object embedded in a file. The client application stores some information for embedded objects, such as the target printer and page position, how the object is activated, and which server application is associated with the object.

The *server application* is any Windows application in which you can edit an object when Windows informs it that you selected the object in a client application. Some server applications also use an *object handler*, which is a dynamic-link library that acts as an intermediary between the client and server applications. The object handler for an application is usually used to improve performance, for example, to redraw a changed object in the window of the client application. Some Windows applications are capable of acting as both OLE client and server.

The discussion in this section uses two other terms: a *source document* is the file where the data or object was originally created; the *destination document* is the compound document where the data is embedded or linked.

Windows Support for Compound Documents

These key Windows elements allow you to take advantage of OLE in Windows 3.1:

- The system registration database, which lists file types and their OLE capabilities. For information about what this database contains and how it is maintained, see "Managing the Registration Database" later in this chapter.
- The associated filename extensions for file types, as defined in the Associate dialog box in File Manager. For information about how to add or change associations between extensions and file types, see Chapter 4, "File Manager," in the *Windows User's Guide.*
- The Object Packager, for creating and editing packages. For information about using the Object Packager, see "Working with Object Packager" in Chapter 13, "Integrating Your Windows Applications," in the *Windows User's Guide*.

Windows 3.1 does not rely on the initialization files to support OLE, although changes in the registration database may be reflected in the **programs=** entry in the **[Windows]**, **[embedding]**, and **[extensions]** sections of WIN.INI.

When you want to create compound documents in Windows, you use several Windows features that you are already familiar with:

- The File Manager, which supports drag-and-drop for printing, embedding data, and other activities that use DDE and OLE features.
- The Clipboard, which has enhanced capabilities in Windows 3.1 to support data from OLE applications.
- The Edit menu commands in Windows applications that support embedding and editing objects.

This section describes how File Manager, the Clipboard, and the Edit menu commands support OLE.

OLE and File Manager

In File Manager for Windows 3.1, if a file is associated with a file type in the registration database, you can:

- Double-click the file's icon in the File Manager window to open the file.
- Drag the file icon to the Print Manager to print the file.
- Drag the file icon to place the file as a group item in any Program Manager group.
- Drag the file icon to an application window to embed or link the file as a packaged object.

Note To create file packages with File Manager, the server application must have been written for Windows 3.1. For applications written for earlier versions of Windows, you must use Object Packager. Also, if the package will contain a linked document, you must use File Manager to place the file in Object Packager.

OLE and the Clipboard

When you choose Cut or Copy in any Windows application that can act as an OLE server, the data on the Clipboard can be used to create an embedded or linked object in any client application.

Clipboard Formats

In Windows 3.1, the data on the Clipboard can include a variety of formats:

- Clipboard data formats, such as **Picture** (for a metafile), **Bitmap**, **DIB Bitmap** (for 256-color palettized graphics), **RTF** (for rich-text format), or **CF_Text** (for Clipboard text format). If you run the Clipboard Viewer after copying data, you can see the list of Clipboard data formats for the data.
- The server application's own formats.
- Native format, which is data that completely defines the object and is interpreted only by the server application or its object handler DLL.
- The OwnerLink format, which identifies the owner and class of a linked or embedded object.
- The Presentation format, which the client application can use to display the object in a document. This can be the Native format if the server application uses an object handler DLL.
- The ObjectLink format (for OLE linking) or the Link format (for DDE linking, as in applications created for earlier versions of Windows). These formats identify the linked object's class and source item and document.

The class name in the ObjectLink or OwnerLink format is a unique name for a class of objects in the server application. These class names are registered in the system registration database. A client application uses this data to identify the owner of an embedded object.

Formats for Pasting from the Clipboard

When you paste data into a destination document, several possible results can occur, based on the format of the data on the Clipboard and the capabilities of the client application:

- If you choose Paste in an application that does not support OLE, then the Native, OwnerLink, and ObjectLink formats are ignored. The application pastes the data in the best Clipboard data format it supports.
- If you choose Paste in an OLE client application, the application checks the Clipboard and places the data in the first acceptable format it finds. For example, if Microsoft Word for Windows finds data in RTF format, it pastes the data that way, rather than creating an embedded object.
- If you choose Paste Special in an OLE client application, the dialog box lists the formats that you can choose for pasting the data as an embedded or linked object.
- If you choose Paste Link in an OLE client application, the application looks for ObjectLink information and makes an OLE link to the source document. If ObjectLink format isn't available, the client application looks for the Link format and creates a DDE link.

When you copy a linked object to the Clipboard and paste it into another destination document, that other document can take advantage of the original link to the source document, because the information associated with a linked object remains with it every time you copy it.

Figure 11.1

Clipboard Viewer

To display this window, in Program Manager click:

OLE and the Edit Menu Commands

A Windows application that supports DDE links and objects has these commands (or similarly named commands) to support placing and editing embedded objects:

- **Copy** copies data from a source document to the Clipboard. A client application can use this data to create an embedded object and may be able to create a link to the source document.
- **Cut** removes data from a source document and places it on the Clipboard. A client application can use this data to create an embedded object.
- **Paste** places data from the Clipboard into the destination document.
- **Paste Link** inserts a DDE link between a document and the file that contains the object. The object appears in the destination document, but the original data that defines the object is stored in the source document.
- **<Class> Object** runs the server application so you can edit or open a link or embedded object. The text for the *<Class>* placeholder depends on the selected object such as Package Object for a selected package. This command name might also be, for example, Edit Object, depending on which actions are allowed for that object. If the object supports more than one action, choosing this command displays a cascading menu that lists other commands.
- **Links** displays the Links dialog box so you can update linked objects, cancel links, repair broken links, or edit or open the linked object.

- **Paste Special** (an optional command) displays a dialog box so you can choose the data format for the object on the Clipboard and also choose either to paste a link or paste the data without making it an object.

Figure 11.2

An example of the Paste Special dialog box, showing possible paste formats

- **Insert Object** (an optional command) displays a dialog box so you can choose which server application to start, then embeds in the destination document the object that is produced by the server. (This is a shortcut for the same result you can get by running the server application, copying the data to the Clipboard, then pasting it in the destination document.)

Figure 11.3

An example of the Insert Object dialog box

Select Package to run Object Packager

After you double-click an object and edit it, you can choose Update from the File menu of the server application to save the changes.

The commands described here are the command names that Microsoft recommends to developers who implement OLE in Windows applications. Check the documentation for your OLE client application to find the actual command names it uses. For examples, see the Edit menus in Write, Paintbrush, and Cardfile.

W

Managing the Registration Database

This section describes how to maintain and modify the system registration database that is installed with Windows 3.1 to support OLE and other activities related to integrating applications.

The registration database (REG.DAT) provides the information that allows you to open or print files by dragging icons in File Manager and Print Manager. The registration database is also used by applications that support OLE, as described in "Windows Support for Compound Documents" earlier in this chapter.

For example, you might want to place a picture from Paintbrush into a Write file. You can do this in two ways, each of which requires information from the registration database:

- You can choose Insert Object from the Edit menu in Write. Write uses the registration database to list types of objects you can insert. When you choose from the list, Write launches the appropriate application for loading or creating that object.
- You can copy a graphic in Paintbrush, then choose Paste Special from the Edit menu in Write. Write uses the registration database to list the names of possible data types for the graphic you want to paste, and lets you choose to embed or link the graphic.

If you later double-click the embedded graphic to edit its contents, Write uses information from the registration database to launch Paintbrush.

When you first install Windows 3.1, the registration database contains information about all the OLE applications included with Windows, plus entries for Microsoft Word for Windows, Microsoft Excel charts and worksheets, Ami Professional, and Professional Write.

Using the Registration Info Editor

If you have an application that supports OLE but is not registered, you can add it to the registration database with the Registration Info Editor (REGEDIT.EXE). However, you will usually not need to use this utility unless your registration database has been corrupted or deleted.

You can use the Registration Info Editor to add, delete, modify, or restore the information that Windows needs to open and print files from File Manager. To do this, choose File Run in Program Manager, then type **regedit** and press ENTER.

If you need help with any item in Registration Info Editor, press F1 to get general help or click the Help button in any dialog box for specific help.

Figure 11.4

Registration Info Editor

To display this window, in Program Manager click:

Note To remain compatible with Windows 3.0 applications that use OLE, Windows 3.1 also adds some registration information to the **[embedding]** and **[extensions]** sections of WIN.INI. However, we recommend that you use the Registration Info Editor to add or change OLE server application information.

Installing a .REG File

If a Windows application has a registration file (.REG), that information is usually added to the registration database when you install the application. But if that .REG file is not merged into the registration database, you can add it with Registration Info Editor.

To install a registration file:

1. Choose Merge Registration File from the File menu in the Registration Info Editor.
2. In the dialog box, select the .REG file for the application to be added to the database, and click the OK button.

Tip You can also double-click a .REG file in File Manager to install that registration file automatically.

Adding a New File Type

You can add a new file type to register an application that doesn't have a .REG file by either copying an existing file type or by creating a new file type.

To copy an existing file type to add registration information:

1. Select a file type from the Registered File Types list in the Registration Info Editor window.
2. Choose Copy File Type from the Edit menu.
3. In the dialog box, modify the registration information for the new file type, specifying a unique Identifier. Then click the OK button.

You cannot change the Identifier for an existing file type.

To create a new file type to add registration information:

1. Choose Add File Type from the Edit menu in the Registration Info Editor.
2. In the dialog box, modify the registration information for the new file type, specifying a unique Identifier. Then click the OK button.

Whichever procedure you use to add a file type, you must specify the following information:

- An Identifier, which must be a unique key word of up to 63 printable ASCII characters. This Identifier is used by Windows to identify the file type in the registration database.
- A File Type, which is the text description that appears in the Registered File Types list in the Registration Info Editor window and in the Associate dialog box in File Manager.
- The action, either Open or Print.
- The command and switches to be executed (or the DDE message to be sent) to perform the action. For example, this command runs the application with the filename being opened or printed (**%1** is the filename parameter):

  pbrush.exe %1

Figure 11.5

Add File Type
dialog box

To display this dialog,
choose Add File Type from the Edit menu in Registration Info Editor

Check the Uses DDE option in the dialog box if the application sends DDE messages to execute the Open or Print actions. The information in the rest of the dialog box specifies the DDE message and application string, and the DDE topic associated with the command.

Modifying and Removing File Types

Use these commands from the Edit menu in Registration Info Editor to either modify or remove a file type from the registration database:

- Choose Modify File Type, and use the dialog box to change the file type information. You cannot change the Identifier.
- Choose Remove File Type to delete any file type from the database.

Restoring the Registration Database

You can restore the registration database if REG.DAT has been deleted or corrupted.

To restore the registration database:

1. Quit Windows, delete REG.DAT, and then restart Windows.
2. Run Registration Info Editor, which will display an empty list of registered file types.
3. Choose Merge Registration File from the File menu.
4. Select SETUP.REG from the File Name list in the dialog box and click the OK button. (This file is in the Windows SYSTEM subdirectory.)

A message appears to confirm that the information has been registered. The database now contains the original registration information that was installed with Windows. If any of your other applications have .REG files, you can add them to the restored database by choosing Merge Registration File from the File menu, as described earlier.

For more information about the related DDE functions and using the Registration Info Editor to create .REG files, see Chapter 7, "Object Linking and Embedding Libraries," in the *Windows Software Development Kit.*

Heller, Martin. “Object Links Coming to Windows Applications.” *Byte* (February 1991): p. 20.