

9

FINDER INTERFACE

Introduction

The Finder

The **Finder** is an application which works with the system software to keep track of files and manage the user's desktop display. On the desktop, the Finder displays **icons** representing your application and the documents it creates. An icon is an image that the Finder displays to graphically represent some object that the user can manipulate.

The Finder needs quick access to some key information about your application, such as which icons to use when displaying your application and its documents. You supply most of this information in the resource fork of your application file. The Finder extracts this information and uses it to maintain its own database of the resources it needs.

The Finder uses certain high-level events known as **Apple events** to communicate certain instructions and information to your application. Accordingly, this chapter should be read in conjunction with the Chapter 10 —Apple Events.

Resources, the Catalog File, and the DeskTop Database

An Application's Required Resources

For compatibility with the Finder, your application should have:

- A **signature resource**, which enables the Finder to identify and start up your application when a user double clicks documents created by your application.
- A set of resources that describe **icons**, which visually represent your application and any documents it creates.
- A set of **file reference resources**, to link icons with the file types they represent, and to allow users to launch your application by dragging the document icons to your application icon.
- A **bundle resource**, which groups together your application's signature, icon and file reference resources.

- A **size resource** (see Chapter 2 — Low Level and Operating System Events) which tells the Finder how much memory to allocate for your application when it starts up and whether your application supports various system software features.

Other resources which should ideally be provided by your application are:

- **Version resources**, which allow users to ascertain the version of your application and, if applicable, the version of your application's superset of files.
- A **finder icon help resource**, which the Finder uses to display your customised balloon help message for your application, control panel, or system extension icon.

The Catalog File

When the user creates or installs a file, the File Manager initially stores some of the information provided by these resources in the volume's **catalog file**. The catalog file is a special file located on the volume which contains information about the hierarchical organisation of files and folders on that volume. Although it is mostly used by the File Manager, the catalog file also contains information used by the Finder.

The DeskTop Database

The Finder extracts from the catalog file the information provided by your resources and, for quick access to that information, uses it to build either a **desktop database** (for all volumes over 2 MB) or a **desktop file** (for volumes under 2MB). The desktop database is a Finder-maintained database of icons, file types, applications, version data, and comments. The desktop file is a resource file in which the Finder stores this information for volumes under 2MB. The Finder updates the database when files are added, moved, renamed or deleted.

The desktop database:

- Contains all icon definitions and their associated file types (see below).
- Lists all the file types that an application can open and all copies or versions of the application listed as the creator (see below) of the file.
- Lists the location of each application on the disk and any comments that the user has added to the **information windows**¹ for desktop objects.

Application Signature, File Creator, and File Types

Application Signature

The Finder identifies your application through its **signature**. A signature is a unique four-character sequence which must not conflict with that of any other application.²

You must include in your resource file a special resource which has your application's signature as its resource type. By convention, the signature resource has a resource ID of 0. The signature resource is typically defined to be a string resource (that is, a resource of type 'STR ') which specifies the name, version number, and release date of your application.³

¹ The window opened when the user selects an icon and chooses Get Info from the Finder's File menu.

² To ensure uniqueness, developers are required to register their application's signature with Apple Computer, Inc, at Macintosh Developer Technical Support.

³ If you do not provide specific version information through a version resource, the Finder displays the string stored in the signature resource in the information window for that file.

Creating an Application Signature Resource Using Resorcerer

Resorcerer allows you to create an application signature resource within its bundle resource editing window (see Fig 5). Fig 1 shows an application signature resource being created using Resorcerer.

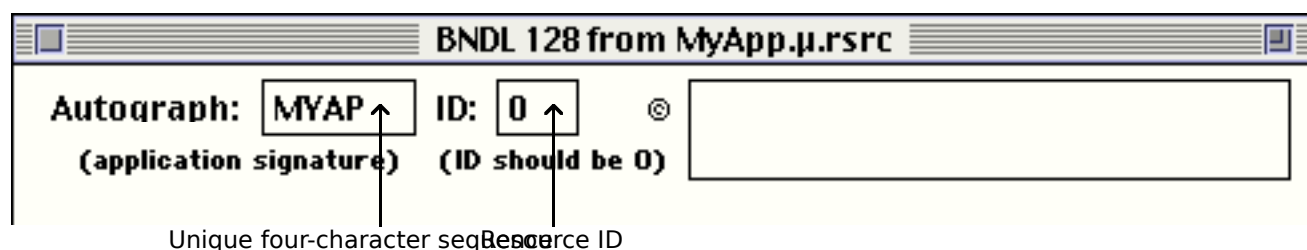


FIG 1 - CREATING A SIGNATURE RESOURCE USING RESORCERER

File Creator and File Type

Whenever your application creates a document, it assigns the document a **creator** and a **file type**. Your application should set its own signature as the document's creator. (The creator field of your application file, incidentally, should contain its own signature.)

Use of the Creator by the Finder

When the user double clicks on a document or selects it and chooses Open or Print from the File menu, the Finder reads the creator field of that file to get the document's creator. The Finder then searches for an application with a signature by that name. When it finds the application, the Finder calls the Process Manager to start your application. The Finder then passes to your application the information it needs to open or print the document via a required Apple event.⁴

File Type

The file type can be a type especially defined for your application or it can be one of the existing general types⁵ shown in the following:

File Type	Description
'APPL'	Launchable application.
'DRVR'	Driver.
'FFIL'	File for storing fonts.
'INIT'	System extension.
'PICT'	QuickDraw picture.
'PRER'	Printer driver.
'RDEV'	Chooser extension.
'TEXT'	Stream of ASCII characters.
'adev'	Network extension.
'appe'	Background only application.
'cdev'	Control panel.
'ffil'	Font.
'ifil'	Script system resource collection.
'kfil'	Keyboard layout.
'pref'	Preferences file.

⁴ Note that in this and other references to your application receiving requests to open and print documents, the assumption is made that your application supports the receipt and handling of required Apple events (see Chapter 10 —Apple Events).

⁵ Apple reserves the use of all signatures and file types whose names contain only lowercase and non-alphabetic characters. Your signature and file type created especially for your application must each contain at least one uppercase character. Like signatures, file types must be registered with Apple.

'qery'	Query document for database access.
'scri'	System extension for script systems.
'sfil'	Sound.
'tfil'	TrueType font.
'ttro'	TeachText read-only file.
'zsys'	A system file (such as the System file).

Your application must have a file type of 'APPL'. The standard file type 'TEXT' should be assigned to files that consist only of text, that is, a stream of characters with return characters at the end of paragraphs. A document of type 'TEXT' can be opened or printed by any application that accepts such file types.

Use of the File Type by the Finder. Another way for a user to open a document created by your application, as well as a document not created by your application but which is of a file type *supported* by your application, is to select its icon and drag the selected icon to your application's icon. Because the document's file type is stored in the catalog file, and the Finder stores a list of your application's supported file types in the desktop database, the Finder can determine whether it should launch your application. If the document's file type is supported by your application, the Finder calls the Process Manager to launch your application and then sends your application a required Apple event containing the information it needs to open the document.

Use of File Type by Your Application. Your application also relies on file types to determine which files to allow the user to open when your application is running. When your application calls the Standard File Package or Navigation Services to open a file, your application supplies either a list of the file types that your application can open or a filter function for those types. The Open dialog box then displays files of the specified types only.

Creating Icons for the Finder

If you do not supply your own icons, the Finder uses its own default application and document icons for display. Fig 2 shows some of the Finder's large and small default colour icons.



FIG 2 - DEFAULT LARGE AND SMALL COLOUR ICONS

However, to distinguish your product for the user, you should design your own icons for all the files associated with your application, including:

- The application file itself.
- Standard documents created by your application.
- Stationery pads that users create from your application's documents. (Stationery pads are files the user creates to serve as templates for other documents.)
- Data-sharing editions that user's create from your application's documents. (Editions are special files that contain data to be shared among applications.)
- Other special documents such as read-only, graphics, and query documents, which are either created by your application or provided by you for use by other applications. (Query documents contain commands and data in a format appropriate for a database or other data source.)

For the most effective display, you should create either an **icon family** or an **'icns' resource** for each of your files.

The Icon Family

An icon family is a set of icons which represent a single object. The following icons make up the icon family for a single file:

<i>Icon</i>	<i>Size (Pixels)</i>	<i>Resource in Which Defined</i>
Large black-and-white icon, and mask	32 by 32	Icon list ('ICN#').
Small black-and-white icon, and mask	16 by 16	Small icon list ('ics#')
Mini black-and-white icon, and mask	12 by 16	Mini icon list ('icm#')
Large colour icon with 4 bits of colour data per pixel	32 by 32	Large 4-bit colour icon ('icl4')
Small colour icon with 4 bits of colour data per pixel	16 by 16	Small 4-bit colour icon ('ics4')
Mini colour icon with 4 bits of colour data per pixel	12 by 16	Mini 4-bit colour icon ('icm4')
Large colour icon with 8 bits of colour data per pixel	32 by 32	Large 8-bit colour icon ('icl8')
Small colour icon with 8 bits of colour data per pixel	16 by 16	Small 8-bit colour icon ('ics8')
Mini colour icon with 8 bits of colour data per pixel	12 by 16	Mini 8-bit colour icon ('icm8')

The small icons appear in the Application menu, and in the Apple menu if the user places the application or an alias to it in the Apple Menu Items folder.

Creating Icon Family Resources Using Resorcerer

Fig 3 shows three icon families being created using Resorcerer, one for the application, one for documents created by the application, and one for stationery created by the application.

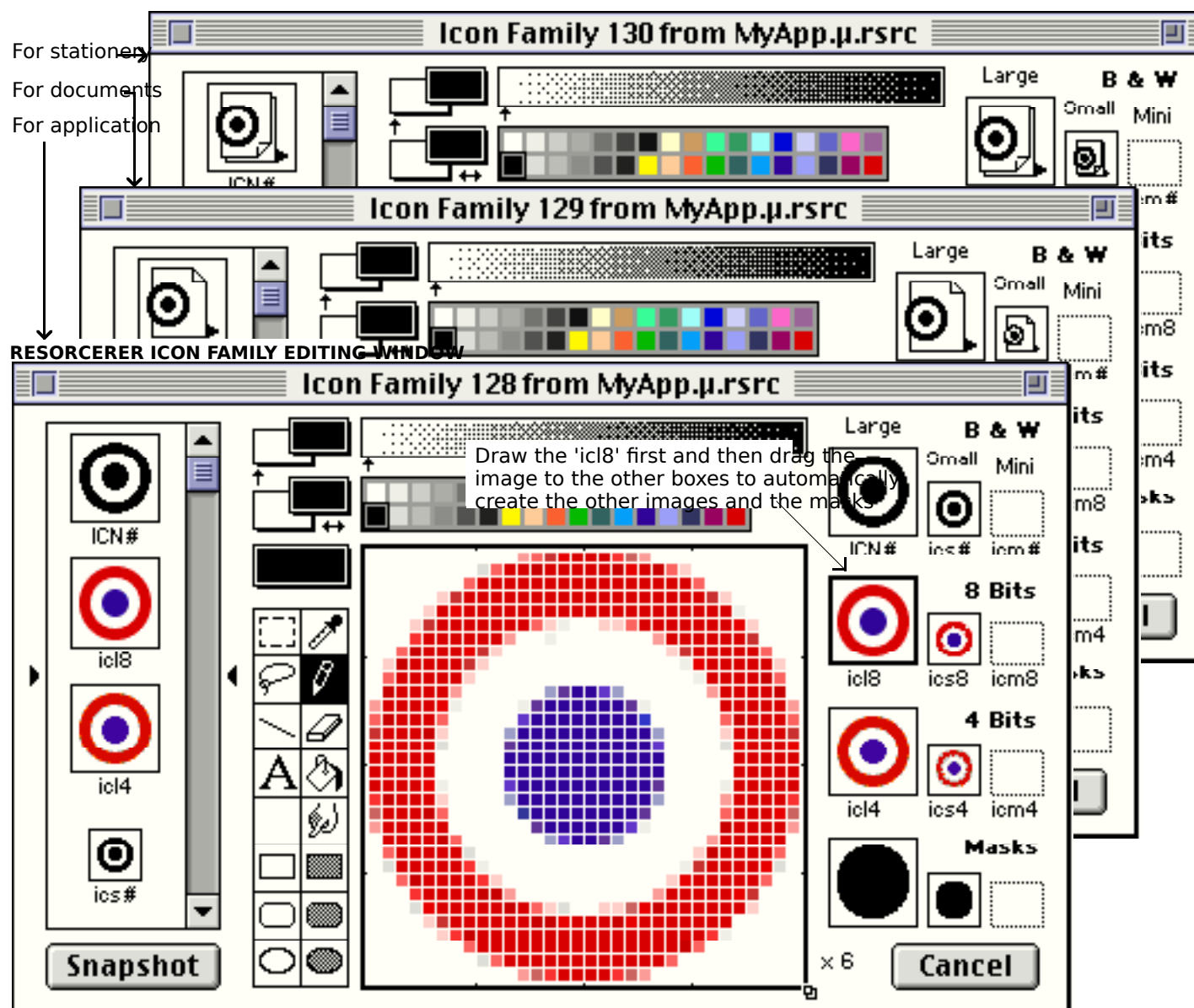


FIG 3 - THREE ICON FAMILIES BEING CREATED USING RESORCERER

The Finder uses the icon's mask (see Fig 3) to crop the icon's outline into whatever background colour or pattern is on the desktop. The Finder then draws the icon into this shape. It is therefore important that the mask be exactly the same shape as the icon. Because the mask also defines the region that users need to click to select the icon, it is best not to have any holes in the icon and thus in its mask.

Good icon design requires that one graphic element be repeated in all icons created for the application. This allows the user to quickly identify the files associated with your product. Also, colours for colour icons should ideally be chosen from the 34 recommended icon colours in the system palette. (See the Apple Icon Colors items in the IconFamily menu in Resorcerer.)

The 'icns' Resource

The 'icns' resource contains all of the data for four icon sizes, thus providing a single source for icon data as opposed to the collection of icon resources ('ics#', 'icl4', 'icm8', etc.) described above. This speeds up icon fetching and simplifies resource management.

Historical Note

The 'icns' resource was introduced with Mac OS 8.5.

The four icon sizes are mini, small, large, and huge, the latter being a new size of 48 by 48 pixels. Four colour depths (1-bit, 4-bit, 8-bit, and 32-bit) and two kind of masks (1-bit and

8-bit) are supported. The deep (8 bit) mask means that masks can have 256 different levels of transparency.

Icon Services checks for an 'icns' resource of the specified ID before it checks for the older resource types ('ics#', 'icl4', 'icm8', etc.) of the same ID. If an 'icns' resource is found, Icon Services obtains all icon data exclusively from that resource.

As of Version 2.2, Resorcerer had no pixel editor for 'icns' resources. However, a command available in the Icon Family editor will take all icons currently being shown and build an 'icns' resource with the same resource ID as those icons. (Choose Build/Update 'icns' from the IconFamily menu.)

Related Resources

If you have defined icon resources for an application and related documents, you must also define file reference resources and a bundle resource for your application (see below).

Customised Document Icons

If, on the other hand, you are simply providing a document which is to be used by other applications, you do not need to create file reference resources or a bundle resource. You can instead create **customised icons** for the document.

Users are able to customise individual icons. By selecting a file and choosing Get Info from the File menu, the user can select the displayed icon and use the Paste command to replace it with a picture from the clipboard. The Finder creates a family of icons (Mac OS 8.1 or earlier) or an 'icns' resource (Mac OS 8.5 or later) based on the user's customised icon, assigns a resource ID number of -16455 to each resource in the icon family, or to the 'icns' resource, stores these resources in the resource fork of the file that the icon represents and sets the hasCustomIcon bit in the file's Finder flags field (see below). You can use the same procedure to provide customised icons for your document.

Note that, although an application can assign icons to all of its documents by associating their icons with the document's file type in a bundle resource, a customised icon can represent only one specific file, that is, that file which has an 'ICN#' or 'icns' resource with an ID of -16455 in its resource fork.

File Reference ('FREF') Resources

File reference ('FREF') resources associate your application's icons with file types created and supported by your application, allowing users to open documents supported by your application by dragging their icons to your application's icon.

You create a 'FREF' resource for your application file itself and you create separate 'FREF' resources for each file type that your application can open.

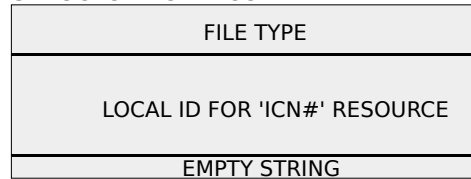
Structure of the 'FREF' Resource and Creating a 'FREF' Resource Using Resorcerer

Fig 4 shows the structure of a 'FREF' resource and a 'FREF' resource being created using Resorcerer. The following describes the main fields of a compiled 'FREF' resource:

<i>Field</i>	<i>Description</i>
FILE TYPE	A four-character code that identifies the type of file represented by this resource.
LOCAL ID FOR 'ICN#' RESOURCE	The Finder uses the local ID number to map the file type specified in this

	resource to an icon list ('ICN#') resource that is assigned the same local ID in the bundle ('BNDL') resource.
EMPTY STRING	This element should always contain an empty Pascal string.

STRUCTURE OF A COMPILED 'FREF' RESOURCE



File type - stationery.

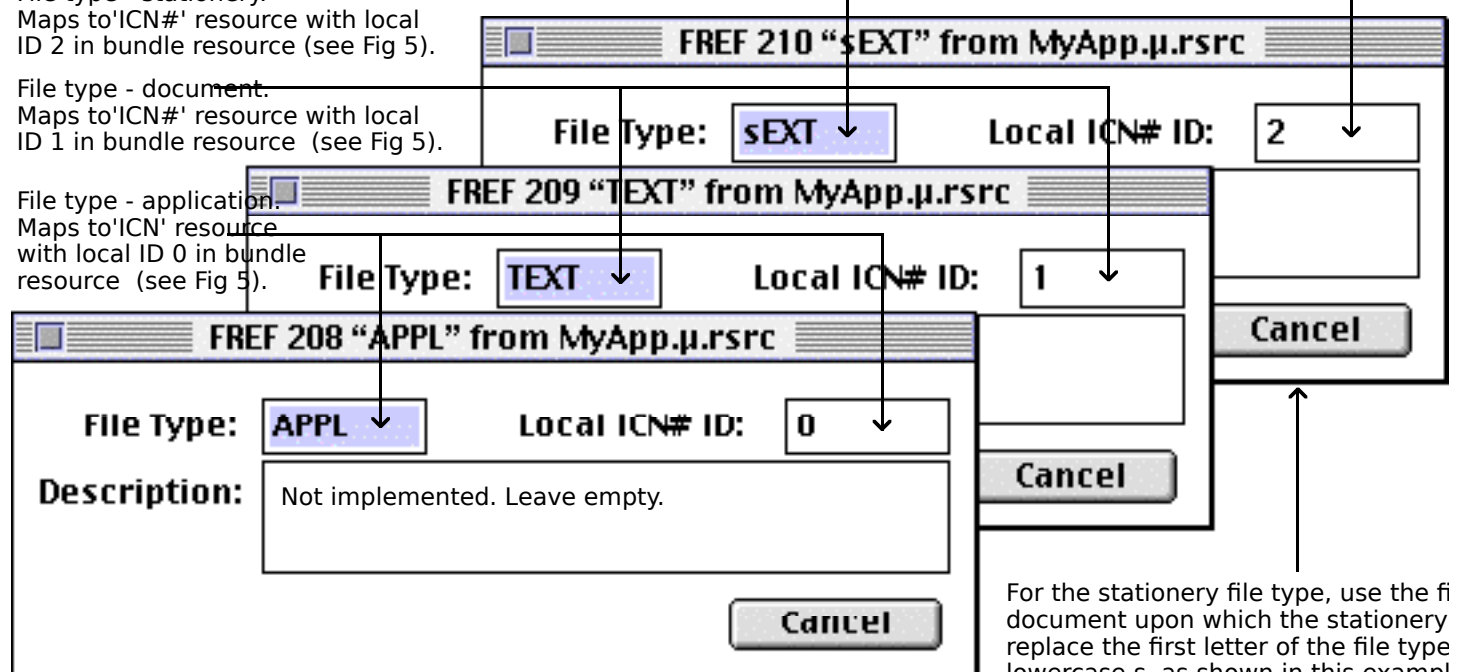
Maps to 'ICN#' resource with local ID 2 in bundle resource (see Fig 5).

File type - document.

Maps to 'ICN#' resource with local ID 1 in bundle resource (see Fig 5).

File type - application.

Maps to 'ICN#' resource with local ID 0 in bundle resource (see Fig 5).



RESOURCE EDITOR 'FREF' RESOURCE EDITING WINDOW

For the stationery file type, use the file type document upon which the stationery replace the first letter of the file type lowercase s, as shown in this example

FIG 4 - STRUCTURE OF A COMPILED 'FREF' RESOURCE AND CREATING A 'FREF' RESOURCE

Use of 'FREF' Resources by the Finder

When the user drags a document icon to your application icon, the Finder checks a list that it maintains of your 'FREF' resources. If the document's file type appears in this list, the Finder launches your application with a request, passed via a required Apple event, to open that document.

If your application supports file types for which it does not provide icons, you can still define 'FREF' resources for them, allowing users to launch your application by dragging these document icons to your application icon. For example, you could create a 'FREF' resource for the document file type 'txt', which is used by documents which have SimpleText as their creator. (The Finder uses SimpleText's icon family resources for these documents.) The user can then open these documents in your application by dragging the document icon to your application's icon.

The Bundle ('BNDL') Resource

A bundle ('BNDL') resource associates all of the resources used by the Finder for your application. In particular, it associates your application and its documents with their icons.

When the Finder displays your application on the user's desktop, it checks the catalog file to see if your application has a 'BNDL' resource. If it does not, the Finder displays the default icons. If it does, the Finder installs the information from the 'BNDL' resource and all its bundled resources into either the desktop database (hard disk) or the desktop file

(floppy disk) and uses this information to display icons for the file types associated with your application.

Structure of the 'BNDL' Resource and Creating a 'BNDL' Resource Using Resorcerer

Fig 5 shows the structure of a 'BNDL' resource and a 'BNDL' resource being created using Resorcerer.

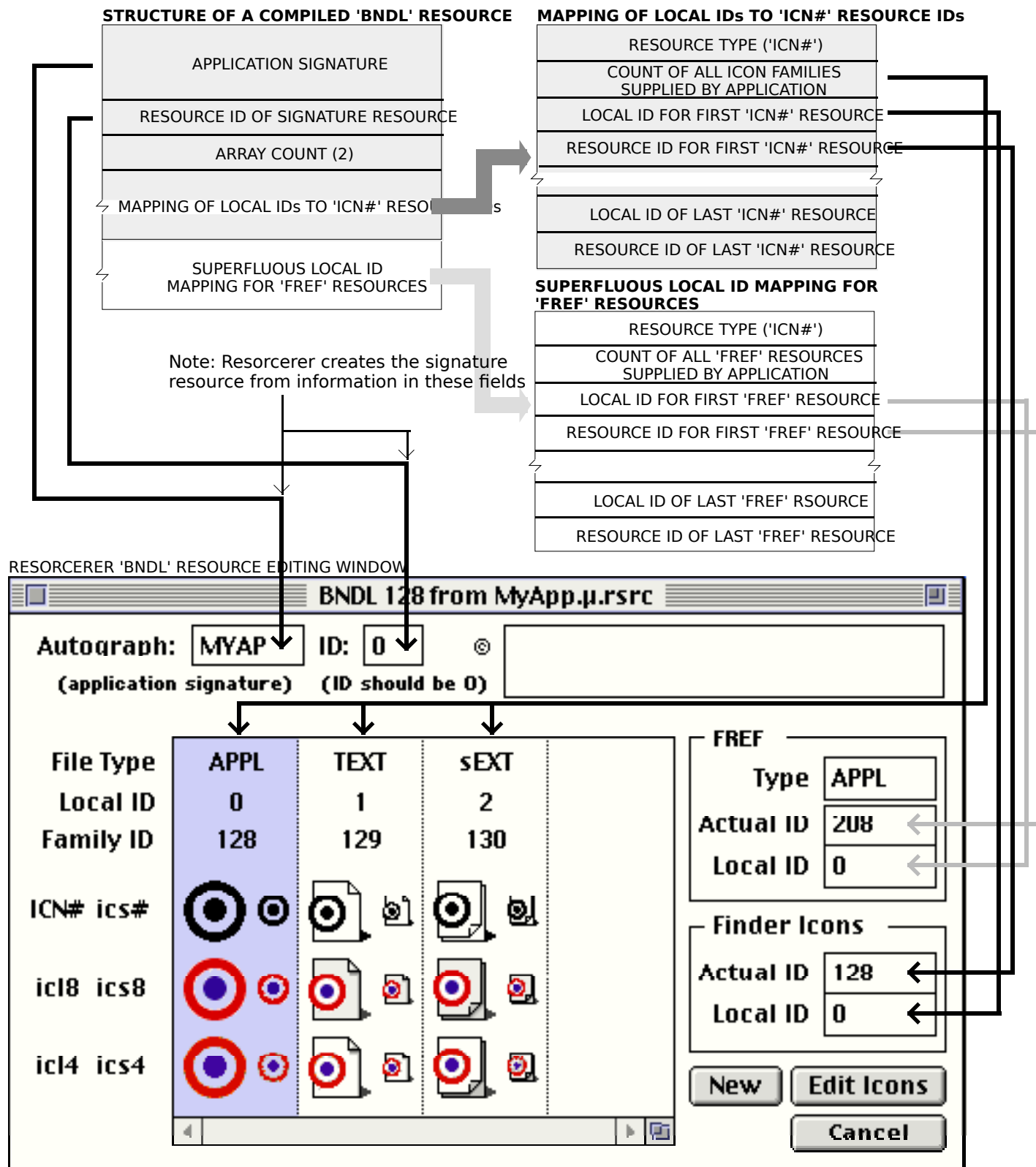


FIG 5 - STRUCTURE OF A COMPILED 'BNDL' RESOURCE AND CREATING A 'BNDL' RESOURCE

The following describes the main fields of a compiled 'BNDL' resource:

Field	Description
APPLICATION SIGNATURE	A unique four-character code that identifies the application to the Finder.
RESOURCE ID OF THE SIGNATURE RESOURCE	The resource ID of the signature resource. By convention, this should always be 0.
LOCAL ID OF FIRST 'ICN#' RESOURCE	Must match the local ID assigned to the 'ICN#' resource within a 'FREF' resource.
RESOURCE ID OF FIRST 'ICN#' RESOURCE	To visually represent files of the type described in the 'FREF' resource that contains the local ID in the preceding field, the Finder uses the appropriate member of the icon family with the same resource ID as this 'ICN#' resource.
LOCAL ID OF FIRST 'FREF' RESOURCE	A local ID for the 'FREF' resource relating to this file type. (Superfluous.)
RESOURCE ID OF FIRST 'FREF' RESOURCE	The resource ID of the 'FREF' resource relating to this file type. (Superfluous.)

Note that you also assign local IDs to 'FREF' resources within the 'BNDL' resource. This assignment is, in fact, superfluous because the Finder does not map these local IDs to any other resources. It was implemented for the earliest versions of the system software, and it remains this way today for backward compatibility.

Resource IDs, Local IDs and the Desktop Database

As shown at Fig 5, you must assign local IDs to your 'ICN#' resources within your 'BNDL' resources. You must ensure that, for all your file types for which you provide icons, these local IDs match the local IDs you assigned inside their corresponding 'FREF' resources.

In the desktop database, the Finder rennumbers the resource IDs that you have assigned to your resources to avoid conflicts with the resources of other applications. Therefore, the 'BNDL' resource has to rely on these local IDs to map 'ICN#' resources to their 'FREF' resources, that is, the 'BNDL' resource uses the local ID you assign to an 'ICN#' resource to map it to the 'FREF' resource that has specified the same local ID. For example, the 'FREF' resource with resource ID 208 at Fig 4 shows that the file type 'APPL' is assigned a local ID of 0. At Fig 5, you will see that local ID 0 is assigned to the 'ICN#' resource with resource ID 128. This maps the icon defined by this resource to the application file.

Fig 6 illustrates how the 'BNDL' resource uses local IDs to map 'ICN#' resources to 'FREF' resources. This figure illustrates two main concepts:

- The one 'BNDL' resource ties together all the icon resources and 'FREF' resources for your application and all its documents.
- The icon resources and their associated 'FREF' resources are mapped together by local IDs.

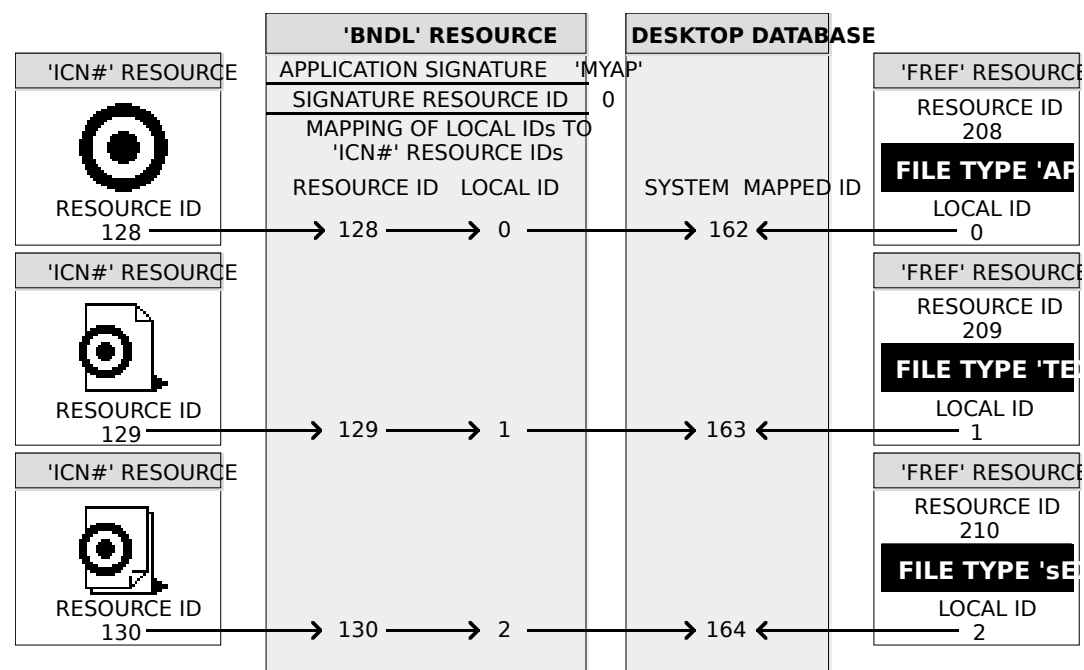


FIG 6 - LINKING 'ICN#' RESOURCES AND 'FREF' RESOURCES IN A

In Fig 6, the application file's 'ICN#' resource has a resource ID of 128 while its 'FREF' resource has a resource ID of 208. For easier code maintenance, you should probably assign the same resource ID to a file's 'FREF' resource that you assign to its 'ICN#' resource.

Informing the Finder that Your Application has a 'BNDL' Resource

You alert the Finder that your application has a 'BNDL' resource by setting a bit in the file's Finder flags field (see below).

Providing Version ('vers') Resources

You can use version ('vers') resources to record version information for your application.

If the window containing your application is the frontmost window and the user chooses as List from the Finder's View menu, then chooses View Options... from the View menu, and then checks the Version checkbox in the View Options dialog box, the version number from the 'vers' resource will appear in the list view against your application's name. The Finder also displays version information when the user selects your application icon and chooses Get Info from the File menu.

You can use 'vers' resources to assign version information to an individual file and, if it is a part of a larger collection of files, to the entire superset of files. The 'vers' resource with ID 1 specifies the version of the file; the 'vers' resource with ID 2 specifies the version of the set of files.

Structure of the 'vers' Resource and Creating a 'vers' Resource Using Resorcerer

Fig 7 shows the structure of a 'vers' resource and a 'vers' resource being created using Resorcerer.

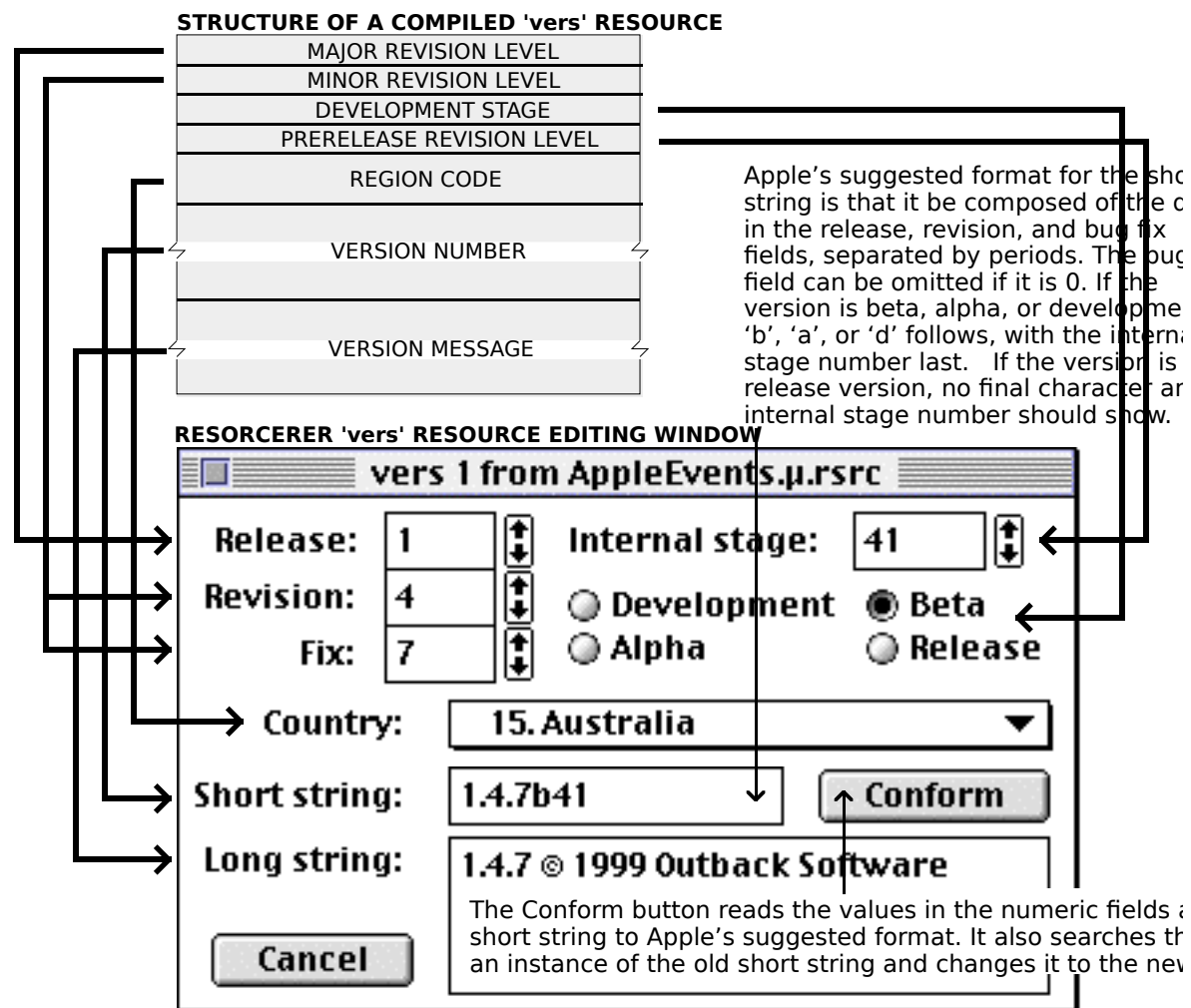


FIG 7 - STRUCTURE OF A COMPILED 'vers' RESOURCE AND CREATING A 'vers' RE

The following describes the main fields of a compiled 'vers' resource:

Field	Description
MAJOR REVISION LEVEL	Major revision level in binary coded decimal format. Although the Finder does not display it anywhere, you can store this information here.
MINOR REVISION LEVEL	Major revision level in binary coded decimal format. Although the Finder does not display it anywhere, you can store this information here.
DEVELOPMENT STAGE	The values in this field indicate one of four development stages: pre-alpha, alpha, beta, or released.
PRERELEASE REVISION LEVEL	The version if the software is still in prerelease.
REGION CODE	Identifies the script system for which this version of the software is intended.
VERSION NUMBER	This Pascal string identifies the version number of the software, and appears in the Version column in list view windows when the Version checkbox in the View Options dialog box is checked.
VERSION MESSAGE	This Pascal string identifies the version number and either a company copyright for a file or a superset of files.

Providing a Finder Icon Help ('hfd') Resource

The Finder provides a default help balloon for your application icon. However, you can provide a customised help balloon by adding a finder icon help override ('hfd') resource with resource ID -5696 to the resource fork of your application.

Fig 8 shows a 'hfd' resource being created using Resorcerer.

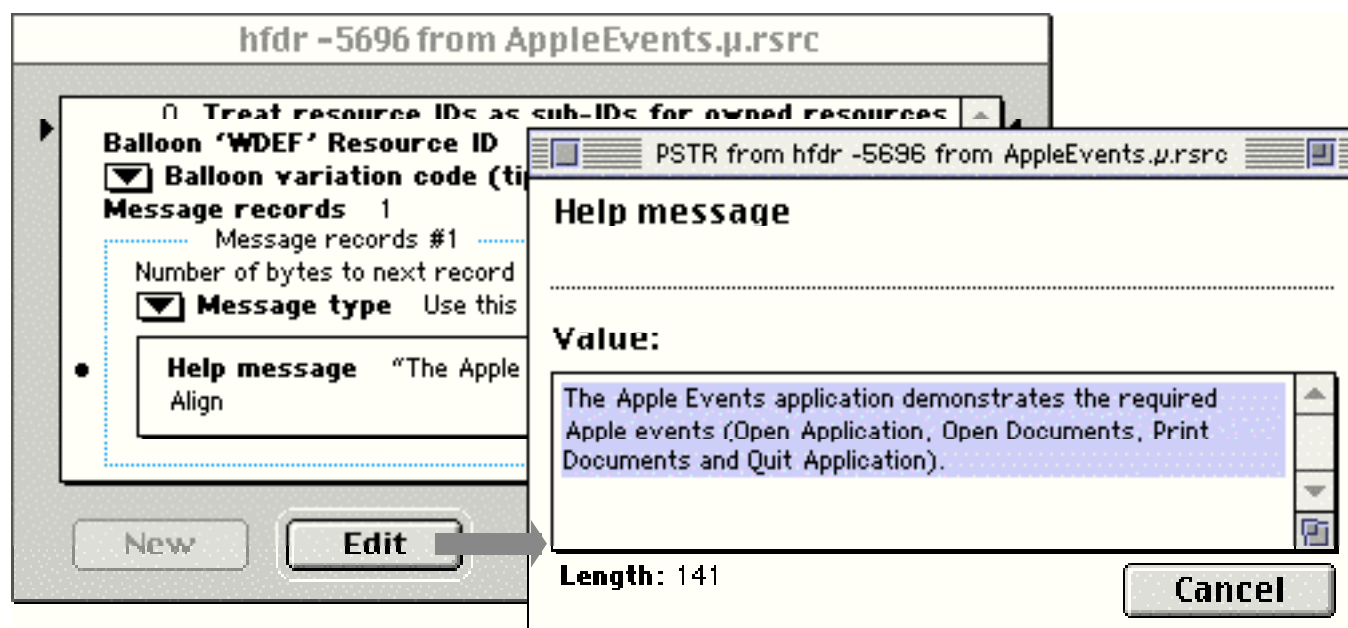


FIG 8 - CREATING AN 'hfdR' RESOURCE USING RESORCERER

How and When the Finder Launches Your Application

User Double-Clicks on the Application's Icon

The simplest scenario under which the Finder launches your application occurs when the user double clicks your application icon or selects it and chooses Open from the Finder's File menu. In these cases, the Finder calls the Process Manager to start your application. The Process Manager creates a partition of memory, loads your code into this partition and sets up the stack, heap and A5 world. The Process Manager then returns control to the Finder.

The Finder then sends your application a required Apple event (called an Open Application event) before relinquishing control to your application. In response to the Open Application event, your application should then perform its usual startup actions, such as opening an untitled document window.

User Double-Clicks a Document Icon or Selects One or More Document Icons and Chooses Open or Print from the Finder's File Menu

The user can request the Finder to open a document created by your application by double-clicking its icon. The user can also request the Finder to open or print documents by selecting one or more document icons and choosing Open or Print from the Finder's File menu.

In these cases, the Finder reads the creator field of each selected file to find the document's creator. If the document's creator matches your application's signature, the Finder calls the Process Manager, which launches your application, and then sends your application a required Apple event (called an Open Documents or Print Documents event) before relinquishing control to your application. These events contain the name, or names, of the document, or documents, to open or print. Your application should then open the documents in titled windows or print them, as appropriate.

User Drags One or More Document Icons to the Application Icon

The user can request the Finder to launch your application by dragging one or more document icons to your application's icon.

In this case, the Finder determines whether to launch your application by comparing the document's file type (stored in the catalog file) against the list of your application's file types. If the document's type appears in the 'FREF' resource list for your application, the Finder calls the Process Manager, which launches your application, and passes your application the name of the selected document, or selected multiple documents, in an Open Documents event. Your application should then open the documents in titled windows.

User Double Clicks a Document Icon - Application Already Running

If the user double clicks a document item while your application is already running, the Finder sends your application an Open Documents event.

Missing Application Name String and Application-Missing String ('STR ') Resources

The Macintosh Easy Open control panel uses translators to open a document created by a program that is not installed on the computer. If Automatic document translation is selected on in the control panel, a translator tells the computer to substitute a similar program. If Always show dialog box is selected on in the control panel, the user will first be presented with a list of applications that might be used to open the document.

The following applies to the situation where the user has selected Automatic document translation off in the Macintosh Easy Open control panel.

Missing Application and Application Missing

If the user tries to open a document created by an application that is missing, or if the user attempts to open a file which he/she should not be able to open (such as a Preferences file), the Finder responds by displaying a modal alert box. The contents of that alert box depend the following factors:

- If the file is a document file created by an application that is missing, whether the resource fork of that file contains a **missing application name string** resource, that is, a 'STR ' resource with ID -16396.
- If the file is a file which is not meant to be opened, whether the file's resource fork contains an **application-missing string** resource, that is, a 'STR ' resource with ID -16397.

If the Finder cannot find the document's creator on any mounted volume, it looks first for the application-missing string resource. If it cannot find an application-missing string resource, it then searches for a missing-application name string resource.

Missing Application

Assuming that the document's name is "Document A", the alert box at Fig 9 is displayed if the file does not contain a missing application name string resource and either the document is not of type 'TEXT' or 'PICT' or the document is of type 'TEXT' or 'PICT' and

SimpleText is not available. If the document is of type 'TEXT' or 'PICT' and SimpleText is available, the alert box at Fig 10 is displayed.

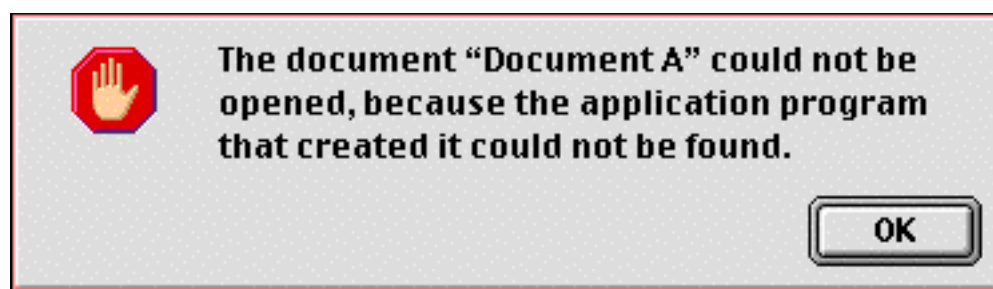


FIG 9 - THE DEFAULT MISSING APPLICATION MODALALER

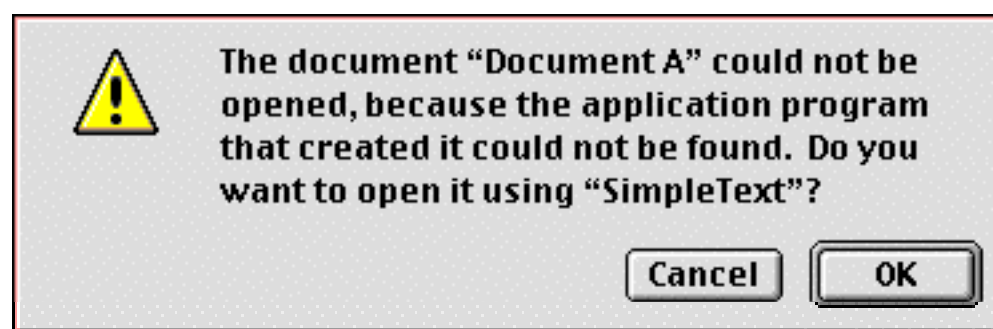


FIG 10 - THE MISSING APPLICATION MODAL ALERT BOX - DOCUMENT AND SIMPLETEXT IS AVAILABLE

Missing Application Name String Resource

The purpose of the missing application name string resource is to provide more specific information to the user. That information is the name of the application that created the program. This is achieved by:

- Including a 'STR' resource containing your application's name in your application's resource fork.
- When your application saves a document for the first time, copying the resource from your application's resource fork to the resource fork of the newly-created document, ensuring that its resource ID is -16396. If this resource is present in the document's resource fork, and the user attempts to open the document when your application is not present, a modal alert box similar to that at Fig 11 is displayed.

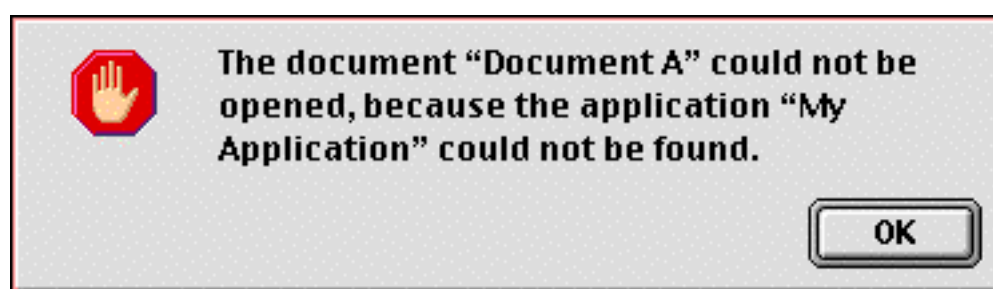


FIG 11 - THE MISSING APPLICATION MODAL ALERT BOX - DOCUMENT APPLICATION NAME STRING RESOURCE

Application-Missing

If the user attempts to open a file that is not meant to be opened, and if the file does not contain an application-missing string resource, the Finder displays a modal alert box similar to that at Fig 9.

Application-Missing String Resource

The purpose of the application-missing resource is to provide more specific information to the user. That information is an explanation of why the file cannot be opened. This is achieved by:

- Including a 'STR' resource containing the explanation in your application's resource fork.
- When your application creates the file, copying the resource from your application's resource fork to the resource fork of the file, ensuring that its resource ID is -16397. If this resource is present in the document's resource fork, and the user attempts to open the file, a modal alert box similar to that at Fig 12 is displayed.

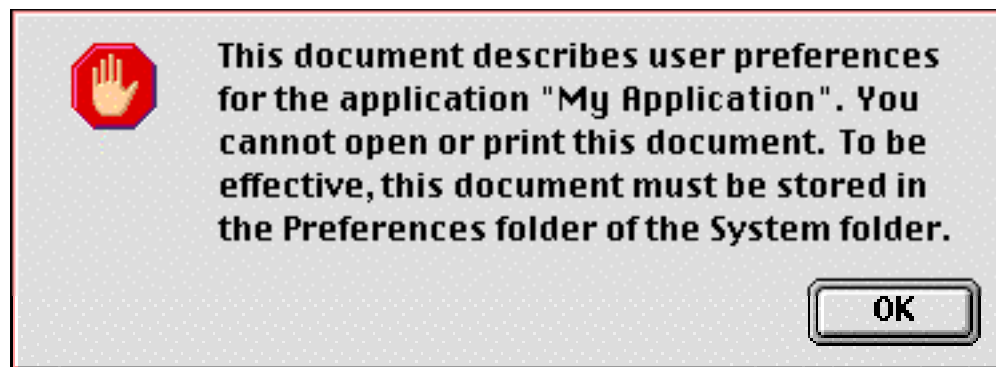


FIG 12 - AN APPLICATION MISSING MODAL ALERT BOX DISPLAYING AN APPLICATION-MISSING STRING RESOURCE

The file must have a unique signature that no other file application known to the finder is likely to have. This ensures that the Finder will display your message, instead of launching the application with that signature, when the user double clicks on the file's icon.

One 'STR' Resource Only

You supply either the missing-application name string resource or the application-missing string resource, but never both. For example, you would supply an application-missing string resource for your preferences file (which the user should not be able to open) and a missing-application name string resource for documents which users should be able to open with your application.

Using Finder Information in the Catalog File

The catalog file exists on all volumes to maintain relationships between the files and directories on that volume. Although it is used mainly by the File Manager, the catalog file is also used by the Finder. The information for files is listed in file information structures (data structures of type `FInfo`) and in extended file information structures (data structures of type `FXInfo`). The information for directories is listed in the directory information (`DInfo`) structures and in extended (`DXInfo`) directory information structures. The Finder manipulates the fields in the file information and directory information structures.

Normally, your application sets the file type and creator information in fields of the file's file information structure when your application creates a new file. For example, the File Manager function `FSpCreate` takes a creator and a file type as parameters⁶. The Finder manipulates the other fields of the file information structure.

The file information structure is as follows:

⁶ See Chapter 16 — Files.


```

struct FInfo
{
    OSType          fdType;          // Type of file.
    OSType          fdCreator;       // File's creator.
    unsigned short  fdFlags;        // Finder flags (invisible, locked, stationery, etc).
    Point           fdLocation;     // File's location in folder.
    short           fdFldr;         // Folder containing file.
};
typedef struct FInfo FInfo;

```

After you have created a file, you can use the File Manager function `FSpGetInfo` to return the file information structure, and you can set the type, creator and flags fields using `FSpSetFInfo`.

Finder Flags

Individual bits of the `fdFlags` field of the file information structure may be examined and set using constants defined in the header file `Finder.h`. The bits of most interest to an application, and the associated constants, are as follows:

Bit	Constant	Value	Description
6	<code>kIsShared</code>	0x0040	Is the file is an application that multiple users on a network can execute simultaneously?
10	<code>kHasCustomIcon</code>	0x0400	Does the file have a customised icon?
11	<code>kIsStationery</code>	0x0800	Is the file a stationery pad? To support stationery pads, your application should check this bit for every document passed to it by either the Finder or the Standard File Package. (<code>StandardGetFile</code> and <code>CustomGetFile</code> return this flag in the <code>sfFlags</code> field of the standard file reply structure.) If the <code>isStationery</code> bit is set for a file the user wants to open, your application should copy the template's contents into a new document and open the document in an untitled window.
12	<code>kNameLocked</code>	0x1000	Can the file be renamed from the Finder and can the file have customised icons assigned to it by users?
13	<code>kHasBundle</code>	0x2000	Does the file have a 'BNDL' resource (which associates the file with its own icons)? When the Finder displays or manipulates a file, it checks the file's <code>hasBundle</code> bit. If that bit is not set, the Finder displays a default icon for that file type. If the <code>hasBundle</code> flag is set, the Finder checks the <code>hasBeenInited</code> bit. If the <code>hasBeenInited</code> bit is set, the Finder uses the information in the desktop database to display the file's icon. If it is not set, the Finder installs the information from the bundle resource into the desktop database and sets the <code>hasBeenInited</code> bit.
14	<code>kIsInvisible</code>	0x4000	Is the file invisible from the Finder and from the Standard File Package dialog boxes?
15	<code>kIsAlias</code>	0x8000	Is the file an alias?

Using the System Folder and its Related Directories

The System Folder is a directory which stores essential system software. The System folder is organised into sub-folders for such files as control panels, fonts, preference files, system extensions, and temporary files. In this way, the Finder keeps the top level of the System folder from being cluttered with dozens, or even hundreds, of files.

Additional folders, such as the Temporary Items and Desktop folders (which are both invisible to the user), are maintained at the root level of the volume.⁷ The Trash directory also exists at the root level of the volume.

You can use the `FindFolder` function to get the volume reference number and directory ID of these folders. **Folder type constants** are passed in the `folderType` parameter of `FindFolder` to specify the folder type.

The only system-related folders you are ever likely to need are:

- The **Preferences** folder, to create and access an application's preferences file. The folder type constant `kPreferencesFolderType` applies. (The demonstration program at Chapter 17 — More on Resources demonstrates creating and accessing a preferences file in the Preferences folder.)
- The **Temporary Items** folder, to create a temporary file. The folder type `kTemporaryFolderType` applies. (The demonstration program at Chapter 16 — Files demonstrates creating a temporary file in, and deleting it from, the Temporary Items folder.)
- The **Trash** folder, to check how much storage is taken by the trash and report this to the user if your application runs out of storage while attempting to save a file. The folder type `kTrashFolderType` applies.

Mac OS 8 introduced, together with a great many new folder type constants, new folder management features to the Finder Interface. Folders can now be added to the System folder, or nested within other folders, and still be located by `FindFolder`. (Previously, `FindFolder` only found folders that were immediately inside the System folder, and a few other folders such as the Trash and Temporary Items folder.) The ability of `FindFolder` to locate folders within folders depends on those folders being registered using the new `AddFolderDescriptor` function, which copies information supplied in a **folder descriptor structure** into the system folder list.

Supporting Stationery Pads

When the user opens a stationery pad from the Finder, the Finder first checks your application's 'SIZE' resource to see if your application supports stationery. If the `isStationeryAware` bit is not set, the Finder creates a new document from the template and prompts the user for a name. The Finder then starts up your application as usual, passing it the name of the new document. If the `isStationeryAware` bit is set, the Finder informs your application that the user has opened a document and passes your application the name of the stationery pad.

To support stationery, your application should specify the `isStationeryAware` constant in its 'SIZE' resource and always check the `isStationeryAware` bit of a document before opening it. The following is an example function which takes a file system specification structure and returns `true` or `false` according to whether the file is a stationery document or not.

```
Boolean isStationeryDoc(FSSpec fileSSpec)
{
    OSErr    osErr;
    FInfo    flInfo;
    Boolean  result;

    osErr = FSpGetFInfo(&fileSSpec,&flInfo);
    if(osErr == noErr)
        result = ((flInfo.fdFlags & kIsStationary) == kIsStationary);
    else
        result = false;
}
```

⁷ The Desktop Folder stores information about the icons which appear on the desktop area of the screen. The user controls the contents of the Desktop Folder by arranging icons on the screen. What appears on the screen is the union of the contents of DeskTop Folders for all mounted volumes.

```

    return(result);
}

```

Unlike the Finder, the Standard File Package always passes your application the stationery pad itself, not a copy of it, regardless of the setting of the `isStationeryAware` bit. When the user opens a stationery pad from within your application, the Standard File Package checks your application's 'SIZE' resource. If the application does not support stationery, the Standard File Package displays an alert box warning the user that the stationery pad itself, not a copy of it, is being opened. This means that the user can thus mistakenly write over the stationery pad by choosing Save without assigning a new name. This problem can be avoided by making your application stationery-aware.

Using Aliases

An **alias** is an object that represents some other file, directory or volume.

Ordinarily, when the user wants to open or print files, your application does not need to be concerned with whether they are aliases because both the Finder and the Standard File Package resolve aliases before passing them to your application. However, if your application opens a file or directory without going through the Finder or Standard File Package, your application should always call `ResolveAlias` just before opening the file.

Relevant Constants, Data Types and Functions

Constants

Finder Flags

<code>kIsOnDesk</code>	= 0x1
<code>kColor</code>	= 0xE
<code>kIsShared</code>	= 0x40
<code>kHasBeenInitd</code>	= 0x100
<code>kHasCustomIcon</code>	= 0x400
<code>kIsStationary</code>	= 0x800
<code>kNameLocked</code>	= 0x1000
<code>kHasBundle</code>	= 0x2000
<code>kIsInvisible</code>	= 0x4000
<code>kIsAlias</code>	= 0x8000

Folder Type Constants

<code>kTrashFolderType</code>	= FOUR_CHAR_CODE('trsh')
<code>kPreferencesFolderType</code>	= FOUR_CHAR_CODE('pref')
<code>kTemporaryFolderType</code>	= FOUR_CHAR_CODE('temp')

Data Types

File Information Structure

```

struct FInfo
{
    OSType          fdType;          // Type of the file.
    OSType          fdCreator;       // File's creator
    unsigned short  fdFlags;        // Flags, for example, hasbundle, invisible, locked, etc.
    Point           fdLocation;     // File's location in folder.
    short           fdFldr;         // Folder containing file.
};
typedef struct FInfo FInfo;

```

Directory Information Structure

```

struct DInfo

```

```

{
  Rect          frRect;          // Folder rect.
  unsigned short frFlags;       // Flags.
  Point         frLocation;     // Folder location.
  short        frView;          // Folder view.
};
typedef struct DInfo DInfo;

```

Functions

```

OSError FSpGetFInfo(const FSSpec *spec, FInfo *fndrInfo);
OSError FSpSetFInfo(const FSSpec *spec, const FInfo *fndrInfo);
OSError ResolveAliasFile(FSSpec *theSpec, Boolean resolveAliasChains, Boolean *targetIsFolder,
                        Boolean *wasAliased);
OSError FindFolder(short vRefNum, OSType folderType, Boolean createFolder, short *foundVRefNum,
                  long *foundDirID);

```

Demonstration Programs

As previously stated, this chapter should be read in conjunction with Chapter 10 — Apple Events. The resources for the demonstration program at that chapter include a signature resource, icon family resources, 'FREF' resources, a 'BNDL' resource, a 'vers' resource, and a 'hldr' resource.

Missing Application Name String and Application-Missing String ('STR ') Resources

The demonstration program at Chapter 16 — Files shows how to copy a missing application name string resource to the resource fork of a document file.

The demonstration program at Chapter 17 — More On Resources shows how to copy an application-missing string resource to the resource fork of a Preferences file.

Using the System Folder and its Related Directories

The demonstration program at Chapter 16 — Files shows how to create a temporary file in, and delete it from, the Temporary Items folder.

The demonstration program at Chapter 17 — More on Resources shows how to create and access a Preferences file in the Preferences folder.