7

Version 1.1

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 **required Apple events** to communicate certain instructions and information to your application. Accordingly, this chapter should be read in conjunction with the Chapter 8 — Required 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 which 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**, 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.

- In documents created by your application, either a **missing-application name string resource** (to display your application's name as part of the text in the alert box which is invoked if the user tries to open or print a document created by your application when the application is missing) or an **application-missing message string resource** (to explain, in an alert box, why the user cannot open or print a file used only by your application — for example, a preferences file).

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 **help resource**, which the Finder uses to display your customised balloon help message for your application, control panel, system extension, or desk accessory 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 typically contains a string 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.

The following is an example signature resource in Rez input format:

```
type 'MYAP' as 'STR ';           /* MYAP is the signature. */
resource 'MYAP' (0, purgeable)   /* Resource ID is 0. */
{
  "My Application 2.0 © 1994"    /* Default Get Info string. */
};
```

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 types[5] shown in the following:

File Type	Description
'APPL'	Launchable application.
'DFIL'	File for storing desk accessories.
'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.
'edtp'	Edition for sharing graphics-oriented data.
'edts'	Edition for sharing sound-oriented data.
'edtt'	Edition for sharing text-oriented data.
'ffil'	Font..
'ifil'	Script system resource collection.
'kfil'	Keyboard layout.
'pref'	Preferences file.
'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).

[4]Note that the use of a required Apple event to communicate this information was introduced with System 7. Another mechanism was used prior to System 7. Note also 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.

[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.

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 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

To distinguish your product for the user, you can 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 an **icon family** for each of your files. 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:

Icon	Size (Pixels)	Resource in Which Defined[6]
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#'`)
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'`)
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'`)

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.

[6]Related to these resources are the icon (`'ICON'`) resource and the colour icon (`'cicn'`) resource. The Finder does not use or display these icons; they are generally used to display an icon in a menu or dialog box.

Default Application and Document Icons. If you do not design your own icons, the Finder uses a set of its own default application and document icons for display. Fig 1 shows the Finder's default large colour icons.

FIG 1 - DEFAULT LARGE COLOUR ICONS

Icon List (`'ICN#'`) Resource

If you do not want the Finder to display the default icons for your application or documents, you must provide, at the least, an icon list (`'ICN#'`) resource for each icon.

If you do not define colour versions of your icons, the Finder displays the black-and-white icon defined in your `'ICN#'` resource on all displays. If you do not define 16 by 16 pixel icons, the Finder algorithmically reduces the 32 by 32 pixel icon to half size when needed.

The Finder uses the icon's mask (which is all black) 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.

An `'ICN#'` resource is defined to be an array of two items of type `String[128]`. Each bit in the first item represents a pixel in the 32 by 32 pixel icon and each bit in the second item represents a pixel in the 32 by 32 pixel mask. The following is a partial listing, in Rez input format, of `'ICN#'` resources that describe the application icon and the icons that represent the documents created by that application.:

```
data 'ICN#' (128, purgeable)   /* Application icon and mask */
{
                               /* [1] APPLICATION ICON */
  $"0E 00 00 00"               /* 1st line of icon: 4 bytes (32 bits). */
  ...                          /* 32 lines total in icon. */
  ,                            /* [2] MASK */
  $"0E 00 00 00                /* 1st line of mask: 4 bytes (32 bits). */
  ...                          /* 32 lines total in mask. */
  ...
};
data 'ICN#' (129, purgeable)   /* Text document icon and mask. */
{
  ...                          /* Icon data */
};
data 'ICN#' (130, purgeable)   /* Stationery pad icon. */
{
  ...                          /* Icon data. */
};
data 'ICN#' (131, purgeable)   /* Edition icon and mask. */
{
  ...                          /* Icon data. */
};
```

Small Icon List (`'ics#'`) Resource

You can also define a small version of your icon in a small icon list (`'ics#'`) resources. On black-and-white monitors, small icons are displayed in windows when the user chooses **by Small Icon** from the Finder's **View** menu. They also 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.

Colour Versions of Large and Small Icons

You should also define colour versions of both large and small icons. The resource for each icon must have the same resource ID as the `'ICN#'` resource that defines the large black-and-white icon. Masks

for the resources which define colour icons are not needed since the colour icons use the masks defined for the black-and-white icons. Colours for colour icons should ideally be chosen from the 36 recommended icon colours in the system palette.[7]

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 based on the user's customised icon, assigns a resource ID number of -16455 to each resource in the icon family, 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). Your 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#'` 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. Example `'FREF'` resources, in Rez input format, are as follows:

```
resource 'FREF' (208, purgeable)  /* My Application. */
{
  'APPL',     /* Type. */
  0,          /* Maps to 'ICN#' resource with local ID 0 in bundle resource. */
  ""          /* Leave empty string for name.  Not implemented. */
};
resource 'FREF' (209, purgeable)  /* My Application document. */
{
  'TEXT',     /* Type. */
  1,          /* Maps to 'ICN#' resource with local ID 1 in bundle resource. */
  ""
};
resource 'FREF' (210, purgeable)  /* My Application stationery pad. */
{
  'sEXT',     /* Type. */
  2,          /* Maps to 'ICN#' resource with local ID 2 in bundle resource. */
  ""
};
resource 'FREF' (211, purgeable)  /* My Application edition */
{
  'edtt',     /* Type. */
  3,          /* Maps to 'ICN#' resource with local ID 3 in bundle resource. */
  ""
};
```

[7]The resource editor ResEdit provides a palette of those colours when you create colour icons using that editor.

```
resource 'FREF' (212, purgeable)
{
  'ttxt',     /* These documents have SimpleText as their creator.  Finder uses */
  4,          /* SimpleText's 'ICN#' resource for these documents.  Included here  */
  ""          /* so users can drag these docs to My Application's application icon. */
};
```

As shown in the example, each `'FREF'` resource specifies the following items:

- The file type.
- The **local ID** of an `'ICN#'` resource as assigned in the bundle resource (see below). The local ID maps the file type to an `'ICN#'` resource that is assigned the same local ID in the bundle resource.[8]
- An empty string.

File Type - Stationery Pad

If you provide your own icon for stationery pads, create a `'FREF'` resource for your stationery pads and assign it a file type in the following manner: use the file type of the document upon which the stationery pad is based but replace the first letter of the original document's file type with a lowercase `s`. (See the `'FREF'` resource with ID 210 in the example.)

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. (See the `'FREF'` resource with ID 212 in the above example .)

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. The `'BNDL'` resource contains:

- The application's signature.
- The resource ID number of its signature resource (which should always be 0).
- The assignment of local IDs to the resource IDs of all `'ICN#'` resources defined for the application. (The local IDs must be the same as those assigned within corresponding `'FREF'` resources.)
- The assignment, for compatibility reasons, of local IDs to `'FREF'` resource IDs. (For consistency, these can be the same local IDs that are assigned inside the `'FREF'` resource, but they do not have to be. They only need to be unique for every `'FREF'` resource.)

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.

---

[8] If you want two file types to share a common icon, you can create two file reference resources that share the same local ID, which the bundle resource will map to the same icon list resource.

Resource IDs and Local IDs for 'ICN#' and 'FREF' Resources

You must assign local IDs to your 'ICN#' resources within your 'BNDL' resources. Make sure 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 file, the Finder renumbers 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 in the previous example shows that the file type 'APPL' is assigned a local ID of 0. In the following example 'BNDL' resource, 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.

The following is an example 'BNDL' resource which relates to the previous example 'FREF' resource:

```
resource 'BNDL' (128, purgeable)
{
  'MYAP',          /* My Application signature. */
  0,               /* Resource ID of signature resource.  Should be 0. */
  {
    'ICN#',        /* Mapping local IDs in 'FREF's to 'ICN#' IDs */
    {
      0,128,       /* 'FREF' with local ID 0 maps to 'ICN#' resource ID 128. */
      1,129,       /* 'FREF' with local ID 1 maps to 'ICN#' resource ID 129. */
      2,130,       /* 'FREF' with local ID 2 maps to 'ICN#' resource ID 130. */
      3,131        /* 'FREF' with local ID 3 maps to 'ICN#' resource ID 131. */
                   /* No 'FREF' with local ID 4 in this list:  TeachText icons
                   /* used for 'ttro' file type. */
    },

    'FREF',        /* Local resource IDs for 'FREF's.  No duplicates. */
    {
      10,208,      /* Local ID 10 assigned to 'FREF' resource ID 208. */
      11,209,      /* Local ID 11 assigned to 'FREF' resource ID 209. */
      12,210,      /* Local ID 12 assigned to 'FREF' resource ID 210. */
      13,211,      /* Local ID 13 assigned to 'FREF' resource ID 211. */
      14,212,      /* Local ID 14 assigned to 'FREF' resource ID 212. */
    }
  }
}
```

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

Other Members of the Icon Family. Of all the icon resource types that make up an icon family, you need to list only the 'ICN#' resource in the 'BNDL' resource. The Finder automatically recognises and loads all other members of the icon family, provided that you have given them the same resource IDs that you have assigned to the 'ICN#' resource.

Fig 2 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.

In Fig 2, 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.

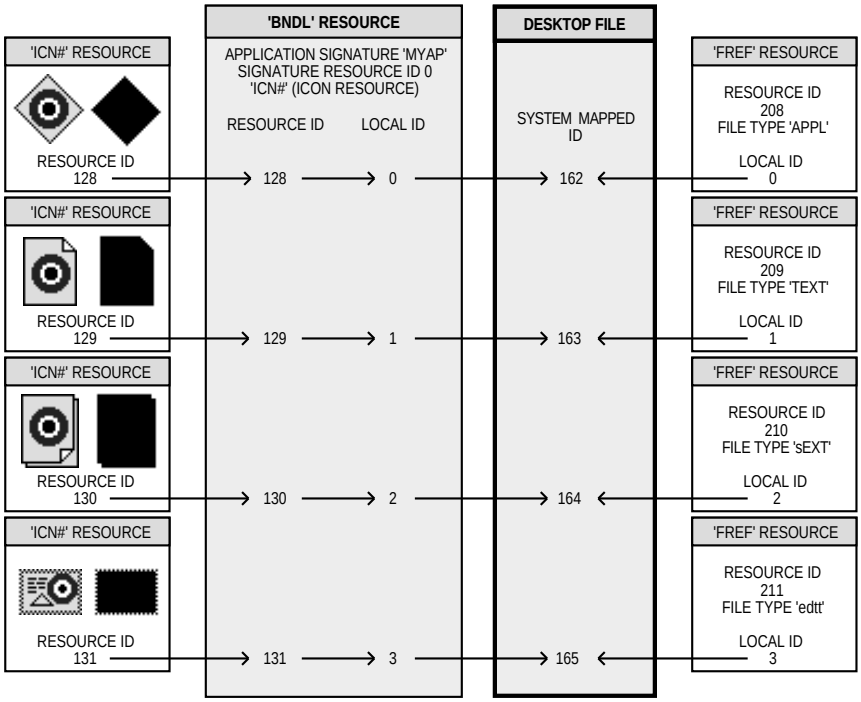

FIG 2 - LINKING ICON LIST RESOURCES AND FILE REFERENCE RESOURCES IN A BUNDLE FILE

Icon Resource for File Types Created by Other Applications

If the user drags a document created by another application to your application icon, and if you have created a `'FREF'` resource for that document's file type, the Finder launches your application and passes it the name of the document. You do not, however, provide icon resources for file types created by other applications because the Finder will use the icon resources defined by the document's creator. Although the local IDs of such a `'FREF'` resource are superfluous in both the `'FREF'` resource and at the bottom of the `'BNDL'` resource, the resource formats nonetheless require that you provide local IDs in both.

For example, note that, in the above `'FREF'` resources example, the `'FREF'` resource with ID 212 is assigned the local ID 4, but that no `'ICN#'` resource is specified in the `'BNDL'` resource. This `'FREF'` resource, which specifies a file type of `'ttxt'`, was created to make the Finder launch the My Application application when the user drags SimpleText read-only documents to the application icon. No icon mapping is made for this file type in the `'BNDL'` resource because the Finder displays the icon defined for it by the SimpleText application. The `'FREF'` resource with ID 212 is assigned the local ID 14 in the `'BNDL'` resource because the format of the resource requires a local ID for all associated `'FREF'` resources.

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). Many development environments provide a simple tool for setting the bundle bit.

Steps Required to Provide Icons for Applications and Documents - Summary

The general steps that you must take to provide icons for applications and documents are summarised as follows:

- Design a graphic element that all of your icon families can share.
- Create an icon list (`'ICN#'`) resource for your application file.

- Create the other members of the icon family for the application file (that is, `'ics#'`, `'icl8'`, `'icl4'`, `'ics8'`, and `'ics4'`) and give each of these the same resource ID as the `'ICN#'` resource.
- Create a `'BNDL'` resource.
- Within the `'BNDL'` resource, list the resource ID number of the application file's `'ICN#'` resource and assign it a local ID of 0.
- Create a `'FREF'` resource for the application file.
- Within the `'FREF'` resource, assign the application a file type of `'APPL'` and assign it the local ID of 0.
- Within the `'BNDL'` resource, list the resource ID number of the `'FREF'` resource for the application file and assign it a unique local ID, for example, 0 to maintain consistency with the local ID assigned in the `'FREF'` resource.
- Create another icon family, consisting of resources of type `'ICN#'`, `'ics#'`, `'icl8'`, `'icl4'`, `'ics8'`, and `'ics4'`, to represent one type of document that your application creates.
- Within the application's `'BNDL'` resource, list the resource ID number of the document's `'ICN#'` resource and assign it a local ID of 1.
- Create a `'FREF'` resource for this document.
- Within the `'FREF'` resource for the document, assign it a file type (for example, `'TEXT'` or `'edtt'`) and assign it a local ID of 1.
- Within the `'BNDL'` resource, list the resource ID number of the `'FREF'` resource for the document and assign it a unique local ID, for example, 1 to maintain consistency with the local ID assigned to the `'FREF'` resource.
- Assigning unique local IDs for every type of document your application creates, repeat the previous five steps.
- If your application supports file types of other applications, define `'FREF'` resources for them, but do not create icons for them.
- Create a signature resource with resource ID 0.
- Set the file's `hasBundle` bit and clear the `hasBeenInited` bit in the file's Finder flags (see below).
- Save and close all the resources.

When you restart your Macintosh, your application should appear with its own icon. If you later alter any of your icons, clear the `hasBeenInited` bit and rebuild your desktop database by pressing Command-Option when restarting.

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

If the user tries to open a document created by an application which 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 an alert box. The contents of that alert box depend the following factors:

- If the file is a document file created by an application which is missing, whether the resource fork of the file contains a **missing application name string** resource.
- 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.
- If system software version 7.5 or later is present and, if so, whether automatic document translation has been selected to on in the Macintosh Easy Open control panel.

The following addresses the Finder's response under System 7 prior to version 7.5, and then describes the change to that response in system software version 7.5 and later.

Finder's Response Under System 7 Prior to Version 7.5

If the user tries to open a document created by your application and your application is missing, the Finder displays an alert box advising the user of the situation. The actual message displayed depends on whether the document is of type `'TEXT'` or `'PICT'`, whether SimpleText or its predecessor,

TeachText, is present, and whether the document file contains a missing application name string resource.

Assuming that the document's name is "Instructions", the alert box at Fig 3 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/TeachText is not available. If the document is of type `'TEXT'` or `'PICT'` and SimpleText or TeachText is available, the alert box at Fig 4 is displayed.

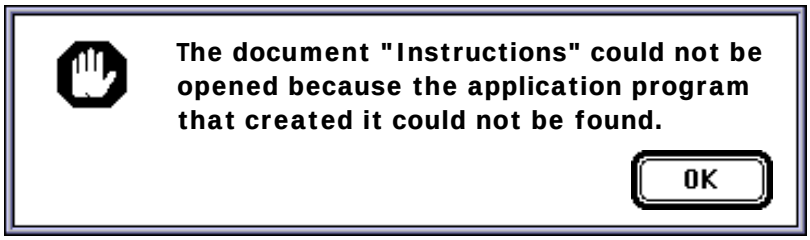

FIG 3 - THE DEFAULT APPLICATION UNAVAILABLE ALERT BOX

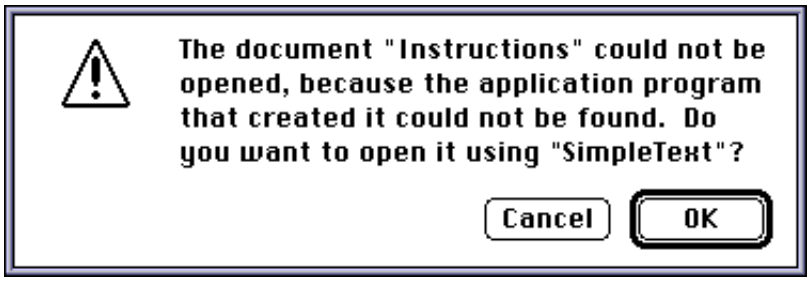

FIG 4 - THE APPLICATION UNAVAILABLE ALERT BOX - DOCUMENT IS A 'TEXT' OR 'PICT' FILE AND TEACHTEXT IS AVAILABLE

Before displaying these default messages, however, the Finder looks in the file for one of two special `'STR '` resources with resource ID numbers of -16397 and -16396. These are, respectively, the **application-missing string** resource and the **missing-application name string** resource. 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.

An application-missing string resource applies to documents which the user should not be able to open. The string is a message explaining why the file cannot be opened. A missing-application name string resource applies to documents which the user should be able to open, assuming your application is present. The string is your application's name.

You supply either the application-missing string resource or the missing-application name 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.

Missing Application Name String Resource

An example of a missing-application name string resource is as follows:

```
resource 'STR ' (-16396, purgeable)
{
  "My Application"
};
```

This resource should be included in your application file's resource fork. Then, when your application saves a document for the first time, it should copy the resource from your application's resource fork to

the resource fork of the newly-created document.[9] 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, an alert similar to that at Fig 5 is displayed.

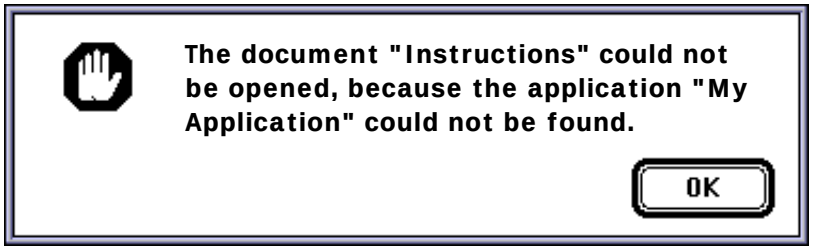

FIG 5 - THE APPLICATION UNAVAILABLE ALERT BOX - DOCUMENT HAS A MISSING APPLICATION NAME STRING RESOURCE

Application-Missing String Resource

Your application-missing string resource should explain why the user cannot open or print the document. Choose a unique signature that is different from your application's signature and any other application's signature and set this signature as the creator of files that you do not want the user to open. This ensures that the Finder displays your message instead of launching the application when the user double clicks on these documents.

The following is an example of an application-missing string resource, in this case for an application's preferences file[10]:

```
resource 'STR ' (-16397, purgeable)
{
  "This document describes user preferences for the application"
  "My Application.  You cannot open or print this document.  To be"
  "effective, this document must be stored in the Preferences"
  "folder in the System folder."
};
```

Finder's Response Under System Software Version 7.5 and Later

Version 7.5 of the system software introduced a new feature called Macintosh Easy Open, which allows Macintosh, DOS and Windows files to be opened even if the programs which created those documents are not present. Available applications, supported by translators, are used to open the documents instead. The Macintosh Easy Open control panel is used to set Easy Open options. Using this control panel, a user may select automatic document translation on or off.

Under system software version 7.5 and later, missing application name string and application missing string resources will cause the Finder to respond as previously described only if automatic document translation is selected to off in the Macintosh Easy Open control panel. If automatic document translation is selected to on, the Finder will invoke the alert box shown at Fig 6. Note that the Finder will invoke this alert box even for files, such as preferences files, which are not meant to be opened.

---

[9]The demonstration program at Chapter 14 —Files shows how to copy a missing application name string resource to the resource fork of a document file.

[10]The demonstration program at Chapter 15 —More on Resources shows how to create a preferences file and copy an application missing string resource to its resource fork.

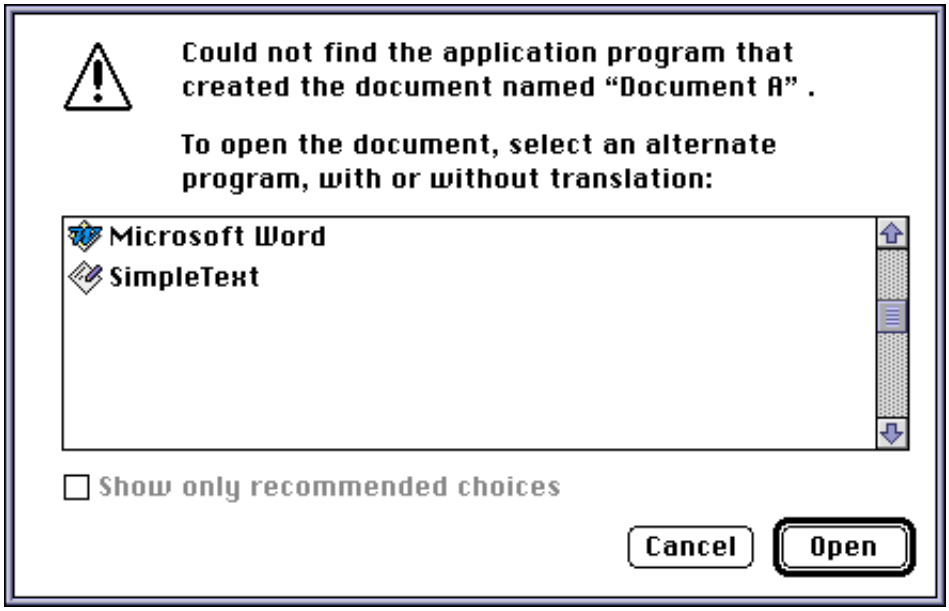

FIG 6 - APPLICATION-NOT-FOUND ALERT BOX — SYSTEM SOFTWARE VERSION 7.5 AND LATER

Providing Version (`'vers'`) Resources

You can use version (`'vers'`) resources to record version information for your application. If the user opens the Views control panel, clicks the Show version checkbox, and then chooses any command from the **View** menu other than **by Icon** or **by Small Icon**, filenames and their version numbers from the `'vers'` resource appear in the active Finder window. The Finder also displays version information when the user selects your application icon and chooses **Get Info** from the **File** menu.

The `'vers'` resource allows you to store a version number, a version message[11], and a region code. 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.

The following is an example of a pair of `'vers'` resources:

```
resource 'vers' (1, purgeable)
{
  0x01,                          /* Major revision level. */
  0x00,                          /* Minor revision level. */
  release,                       /* Development stage. */
  0x00,                          /* Prerelease revision level. */
  verUS,                         /* Region code. */
  "1.0",                         /* Version number */
  "1.1 (US), © My Company 1994" /* Version message */
};
```

[11]Because the **Get Info** command's information window already displays the name of your application, the version message should not include the name of your application.

```
resource 'vers' (2, purgeable)
{
  0x02,                         /* Major revision level. */
  0x00,                         /* Minor revision level. */
  release,                      /* Development stage. */
  0x00,                         /* Prerelease revision level. */
  verUS,                        /* Region code. */
  "2.0",                        /* Version number */
  "(for My Application)         /* Version message */
};
```

`Major Revision Level` The major revision level is in binary coded decimal format. Although the Finder does not display it anywhere, you can store this information here. Most programming environments provide a tool for setting this element.

`Minor Revision Level` The minor revision level is in binary coded decimal format. Although the Finder does not display it anywhere, you can store this information here. Most programming environments provide a tool for setting this element.

`Development Stage` Use any of these values or the constants which represent them:

Value	Constant	Description
`0x20`	`development`	Prealpha file
`0x40`	`alpha`	Alpha file
`0x60`	`beta`	Beta file
`0x80`	`release`	Released file

`Prerelease Revision Level` The version number if the software is still in prerelease.

`Region Code` Script system for which this version of the software is intended.

`Version Number` Version number of the software. The Finder window containing this application displays this string if Show version is selected in the Views Control panel and anything other than **by Icon** or **by Small Icon** is chosen from the **View** menu.

`Version Message` Identifies the version number and either a company copyright for a file or a product name for a superset of files. When the user selects this file, and chooses the **Get Info** command, the Finder displays this string in the information window as follows:

- For a `'vers'` resource with ID 1, in the version field.
- For a `'vers'` resource with ID 2, beneath the file's name next to the file's icon.

Fig 7 shows how the Finder displays the information from these resources in the information window.

You can store `'vers'` resources in any kind of file, not just an application.

As previously stated, if your application does not contain a `'vers'` resource with ID 1, the Finder displays the string from your signature resource as the version information in the information window for your application.

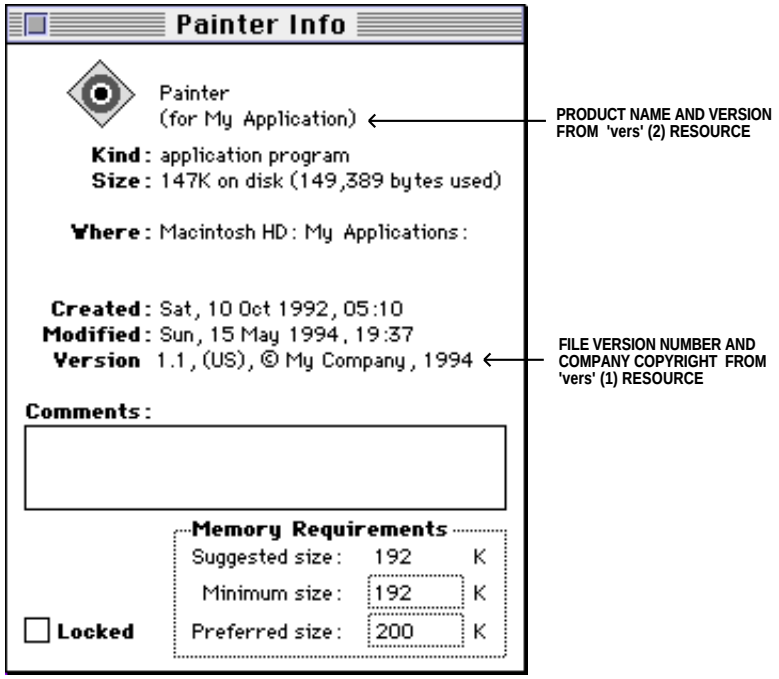

FIG 7 - VERSION DATA IN THE INFORMATION WINDOW

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 records (data structures of type `FInfo`) and in extended file information records (data structures of type `FXInfo`). The information for directories is listed in the directory information (`DInfo`) records and in extended (`DXInfo`) directory information records. The Finder manipulates the fields in the file information and directory information records.

Normally, your application sets the file type and creator information in fields of the file's file information record when your application creates a new file. For example, the File Manager function `FSpCreate` takes a creator and a file type as parameters[12]. The Finder manipulates the other fields of the file information record.

The file information record is as follows:

```
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 record, and you can set the type, creator and flags fields using `FSpSetFInfo`.

[12]See Chapter 14 — Files.

Finder Flags

Individual bits of the `fdFlags` field of the file information record 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	`kIsShared`	`0x40`	Is the file is an application that multiple users on a network can execute simultaneously?
10	`kHasCustomIcon`	`0x400`	Does the file have a customised icon?
11	`kIsStationery`	`0x800`	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. (`StandardGetFile` and `CustomGetFile` return this flag in the `sfFlags` field of the standard file reply record.) If the `isStationery` 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	`kNameLocked`	`0x1000`	Can the file be renamed from the Finder and can the file have customised icons assigned to it by users?
13	`kHasBundle`	`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 `hasBundle` bit. If that bit is not set, the Finder displays a default icon for that file type. If the `hasBundle` flag is set, the Finder checks the `hasBeenInited` bit. If the `hasBeenInited` 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 `hasBeenInited` bit.
14	`kIsInvisible`	`0x4000`	Is the file invisible from the Finder and from the Standard File Package dialog boxes?

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 record and returns `true` or `false` according to whether the file is a stationery document or not.

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

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

  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.

Providing Balloon Help for Non-Document Icons

The Finder provides default help balloons for application, control panel and system extension icons. You can provide a customised help balloon by adding a help override 'hfdr' resource with resource ID -5696, together with its associated 'STR ' resource, to the resource fork of your application. The following is an example:

```
resource 'hfdr' (-5696, purgeable)
{
  HelpMgrVersion, hmDefaultOptions, 0, 0, /* Header information. */
  {HMSTRResItem {kIconHelpString}}
};
resource 'STR ' (kIconHelpString, purgeable)
{
  "Use the My Application word processor to create and edit documents."
};
```

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.

Using the System Folder and its Related Directories

The System Folder is a directory which stores essential system software such as the System file, the Finder, and printer drivers. The Finder maintains a Temporary Items folder and a DeskTop folder[13] at the root level of the volume, both of which are invisible to the user in that they do not appear in the System Folder window. The Trash directory also exists at the root level of the volume.

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

- Preferences, to check for the existence of an application's preferences file.
- Temporary Items, to create a temporary file.
- The trash, 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.

Note that you cannot be certain of the location of system-related directories such as Preferences. In future versions of the system software, they may not be located in the System Folder. Accordingly, you should always use FindFolder to get the path information to these directories.

[13]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.

Relevant Constants, Data Types and Routines

Constants

Finder Flags

```
kIsOnDesk       = 0x1
kColor          = 0xE
kIsShared       = 0x40
kHasBeenInited  = 0x100
kHasCustomIcon  = 0x400
kIsStationary   = 0x800
kNameLocked     = 0x1000
kHasBundle      = 0x2000
kIsInvisible    = 0x4000
kIsAlias        = 0x8000
```

Data Types

File Information Record

```
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 Record

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

typedef struct DInfo DInfo;
```

Routines

```
OSErr    FSpGetFInfo(const FSSpec *spec,FInfo *fndrInfo);
OSErr    FSpSetFInfo(const FSSpec *spec,const FInfo *fndrInfo);
OSErr    ResolveAliasFile(FSSpec *theSpec,Boolean resolveAliasChains,Boolean *targetIsFolder,
         Boolean *wasAliased);
OSErr    FindFolder(short vRefNum,OSType folderType,Boolean createFolder,short *foundVRefNum,
         long *foundDirID);
Boolean  BitTst(const void *bytePtr,long bitNum);
```

Demonstration Program

As previously stated, this chapter should be read in conjunction with Chapter 8 — Required Apple Events. The resources for the demonstration program at that chapter include a signature resource, an icon family, `'FREF'` resources, a `'BNDL'` resource, a `'vers'` resource, and a `'hfdr'` resource.

The demonstration program at Chapter 14 — Files shows how to copy a missing application name string resource to a document file. The demonstration program at Chapter 15 — More on Resources shows how to copy an application missing string resource to a preferences file.

Creating Finder Interface Resources Using ResEdit

A description of how to create a signature resource, an icon family, `'FREF'` resources, a `'BNDL'` resource, a `'vers'` resource, and a `'hfdr'` resource is at Chapter 8 — Required Apple Events.