11-17-1988

Application Calling

Michael E. Hussey
Purdue University

John C. Lindenlaub
Purdue University

Follow this and additional works at: https://docs.lib.purdue.edu/ecetr

https://docs.lib.purdue.edu/ecetr/642

This document has been made available through Purdue e-Pubs, a service of the Purdue University Libraries. Please contact epubs@purdue.edu for additional information.
Application Calling*

Michael E. Hussey
John C. Lindenlaub

TR-EE 89-10
November, 1988

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

* Supported in part by grants from the Westinghouse Education Foundation and the AT&T Foundation
Application Calling*

Michael Hussey
Student Intern

Professor J. C. Lindenlaub
Faculty Sponsor

School of Electrical Engineering
Purdue University
West Lafayette, Indiana 47907

November 17, 1988

* Supported in part by grants from the Westinghouse Education Foundation and the AT&T Foundation
Introduction

The motivation for the development of Application Calling grew from an interest in offering a large number of users access to software on an AppleTalk network. The goal of Application Calling was to give access to applications and yet:

- limit the disk space requirements
- protect software vendors from unauthorized copying of software
- maintain the standard Macintosh interface
- allow for easy software updates

Application Calling does this by using a different method of accessing software. Instead of every user having a copy of an application, every user has a copy of a small file which launches or "calls up" the real application. The real applications are then hidden so that they cannot be easily copied.

There are, however, two disadvantages to Application Calling:

- it can be difficult to implement because the applications must initially be altered using a special application
- some applications may not be able to be altered properly

The following paper is divided into two major parts. The first part presents ways to use Application Calling, and the second explains the technical points of setting it up.
Part I. How to Use Application Calling

In order to use Application Calling, it is necessary to understand a little more about it. Consider the word processing program MacWrite. There are three familiar files associated with this application. Their icons appear on the desktop as shown in Figure 1.

![MacWrite Icons](image1)

Figure 1. The Three Familiar MacWrite Icons

The first file is the application, the second is a regular MacWrite document, and the third is a text-only document. If a user double clicks on any one of the icons (or chooses Open from the File menu), MacWrite is launched. In the case of the last two, the documents are also opened.

If MacWrite has been altered so that Application Calling can be used, there are four files associated with it instead of three. Their icons appear on the desktop as shown in Figure 2.

![MacWrite Icons](image2)

Figure 2. The Four MacWrite Icons Using Application Calling

The first three files are exactly the same as the ones just mentioned. The fourth one is the special "calling" file. If a user double clicks on its icon, MacWrite is launched, but the file is not opened. This is exactly the same thing that happens when a user double clicks on the actual application's icon. Because these two files, MacWrite and MacWrite.CAll, are so much alike, it is difficult for the average user to tell them apart. The only obvious difference between them (besides their names) is their size: the application is very large, whereas the "calling" file is very small.

All of the applications protected by Application Calling have this extra file associated with them. Two examples will now be presented of how these "calling" files are used with a server.

1. Using With a Server That Stores Both the Users' Files and Applications

As an example, consider the application MacWrite being protected on a server running AppleShare. Assume that the server has the folders and files on it as shown in Figure 3. All of the purchased copies (three in this example) of the MacWrite application are in their own folders.
and all of these folders are inside the rap folder (rap is short for "real applications.") A copy of the small "calling" file is in the applications folder. (Note that its name is MacWrite, not MacWrite.Call. The reason for this will be explained later.)

![Figure 3. A Server With Both the Users' Files and Applications](image)

If a user double clicks on the "calling" file in the applications folder, then one of the copies of MacWrite is found and launched. Now suppose that the rap folder is thrown away. The desktop still looks as if a copy of MacWrite is in the applications folder. Yet, if a user double clicks on this "calling" file, then the real MacWrite cannot be found and an appropriate error message results. There is, however, a way to make the rap folder invisible to the desktop without throwing it away. (This method is described in detail in Part II.)

So the rap folder is not thrown away but instead made invisible. With the rap folder invisible, it looks like there is only one copy of MacWrite on the server and it is in the applications folder. But this time, however, if a user double clicks on it, then MacWrite will be launched. This is because the real applications are on the server, but they just cannot be seen from the desktop.

A copy of this small file in the applications folder is placed in each of the four users' folders, Bob, Joe, Sue, and Bud. It looks as if there are now five copies of MacWrite on the server, but in actuality there are precisely three and they are hidden inside the rap folder.

Assume that there are four Macintoshes which are connected to the server through the
Four users, Bob, Joe, Sue and Bud, all sit down at a Macintosh, log onto the server and open up their own folders. All of them can double click on their "calling" files and run MacWrite, but only three of them at a time. This is because once three of them are running MacWrite, the three copies of the application in the rap folder are busy and therefore inaccessible to the fourth user.

This example can be expanded to include other applications and more users. The purchased or licensed number of copies of a particular application are placed in the numbered folders inside the rap folder. The "calling" file is placed in the applications folder so that every user can get a copy of it. It does not matter how many users then log onto the server and run an application, because only the number of copies of it in the rap folder can be launched simultaneously. If someone decides to take a copy of the "calling" file to a Macintosh that is not connected to the file server, the unethical fellow will not be able to launch the application. This is because he no longer has access to the real applications which are stored in the rap folder.

2. Using With a Server That Does Not Store Users' Files

In this kind of server configuration, each user keeps his personal files on some other medium such as the startup disk, other servers, or hard drives. This kind of server configuration is especially useful when there are hundreds of users and not very much disk space on the server. As an example, again consider MacWrite being protected on a server running AppleShare. Assume that the server has the folders and files on it as shown in Figure 4. All of the purchased copies (three in this example) of the MacWrite application are in their own folders inside the rap folder. A copy of the small "calling" file is in the applications folder.
The **rap** folder is made invisible, leaving only two folders, **applications** and **user info**, to appear on the desktop. If a user logs on, he can open the **applications** folder and see the MacWrite "calling" file. He can also open up the **user info** folder to find certain kinds of information, such as instructions on how to operate MacWrite. It would appear to him that only one copy of MacWrite is on the server and he would see no personal folders as in the last example. The existence of only a few folders gives the user the impression that no one else is using the server except himself. However, if another user logs on, he sees the same thing. Both of them can look in the **user info** folder and both can go inside the **applications** folder and double click on the MacWrite "calling" file. (Note that this file is not opened by either user so neither should get a file busy error. However, if they double click on the "calling" file at exactly the same time, there is a possibility of one receiving a busy error.)

As with the previous example, this one can be expanded to include other applications and more users. The other applications are simply placed in the numbered folders inside the **rap** folder and their "calling" files in the **applications** folder.

**Important Server Questions**

How many copies of a given application should go on the server? This decision needs to
be made based on the availability of disk space, the cost of licensing copies, and the importance
of the user always being able to always launch the application. As long as the number of copies
of an application equals the number of Macintosches connected to the server, everyone will
always be able to run it. If, however, there are fewer copies, then there is chance that a user will
not be able to launch it whenever he wants to.

How are software updates handled? Simply by keeping the most recent versions of the
applications in the rap folder, all of the users will be running the newest software.

What sort of privileges and groups are needed when using Application Calling?* The
privileges of applications, rap, and the numbered folders inside rap should be read only to all
users. This prevents them from accidentally throwing away or saving information in these
folders. The top directory should also be read only so that the users cannot change the
invisibility of the rap folder. It is suggested that every user be made a member of a group like
"users" and then setting the group privileges of the above mentioned folders to read only for the
"users" group.

Privileges can also be used to give certain users sole access to some software. This is done
by putting the software in a folder with special group privileges. Then, only those users who are
members of this group can launch the software. This is especially useful if only a limited
number of copies of an application are available.

Why use the configuration outlined in the second example? The advantage of this
arrangement, is that it allows a very large number of people to access applications without them
having to get personal accounts on the server. There could be one general account which simply
gives the user read only privileges. Those people who want access to the applications only need
to obtain this user name, its password, and access to a Macintosh which is connected to the
server. This can simplify things for both the user and the server administrator. An arrangement
that uses both of the mentioned configurations could also be implemented.

How does Application Calling save disk space? It only saves disk space if the number of
users is larger than the number of copies of an application. For example, if there are 100 users
who each use a 500K software package on a 20 Macintosh network, Application Calling would
save 39.9 megabytes [ (100 users x 500K) - (20 copies x 500K) - (100 users x 1K) ]. When
using Application Calling, the general formula for the approximation of the disk space saved for
each application is:

\[
\text{[(number of users) - (number of copies in rap)] x (size of the application)}
\]

*See the AppleShare Administrator's Guide for more details on users, groups, and privileges.
In order for someone to set up Application Calling he will need:

- a familiarity with the Macintosh
- floppy disks that can be used exclusively for testing the applications
- a basic knowledge of the applications to be protected
- a copy of ResEdit, the Apple Macintosh resource editor (see Appendix I for more information on this application)

The setting up of Application Calling involves four stages:

A. Creating the "calling" files
B. Making the "calling" files look like the applications
C. Moving the applications to the server
D. Making the rap folder invisible.

It should be pointed out that a complete understanding of how all of these steps work is not necessary for Application Calling to be successfully used. The following explanations are brief and intended primarily to give the basic ideas behind each stage. General step-by-step procedures are given for completing the more complicated stages so that an in-depth understanding is not required.

A. Creating the "Calling" Files

There are two types of files on the Macintosh; they are called applications and documents. Generally, applications contain program code and documents contain data. Every file has a header in which there are, among other things, the two fields, file type and creator. (See Figure 5.) These two fields contain a four byte unique character string known as a signature. These fields are used by the Finder to determine a file's type and, in the case of a document, the application that created it. When a user double clicks on a file's icon, the Finder looks at the file's header to see if it is an application (file type "APPL") and launches it if it is. If the file is not an application, then the Finder tries to locate the application with the same creator field as the file's.
Figure 5. Part of a Header from a MacWrite Document

As an example, look carefully at Figures 5 and 6. The header of the MacWrite application shows that the file type is "APPL" and the creator is "MACA." If a user double clicks on the application's icon, the Finder, seeing that its file type is "APPL", knows that the file is an application and will launch it. If a user double clicks on the document's icon, the Finder looks at the file type, sees that it is not an application, and therefore tries to launch the application that has the same creator field as the document, namely "MACA."

Figure 6. Part of a Header from the MacWrite Application

Knowing these few things about Macintosh files, it is possible to see how to create the "calling" file that is used in Application Calling. A document needs to be made whose creator field is the same as that of an application's. The application can then be launched by double clicking on that document's icon.

General Procedure to Make the "calling" File for an Application:

1) Launch ResEdit. (See Appendix I for information concerning the use of ResEdit.) Find and select the application that is to be used with Application Calling. Choose Get Info from the File menu.

2) Write down the signature which is in the Creator field. (Case is important.) This signature is the application's creator type. Close the window.

3) Choose New from the File menu. In the New file name field, type in the name of the application plus a "Call" suffix. For example, if the application is MacWrite, the name should be "MacWrite.Call". Click on OK. A window appears whose title is the name just entered. Close this window and the "calling" file is created.

4) If the file with the "Call" suffix is not already selected, do so. Get Info on this file.
Press the Tab key two times to move to the **Creator** field. (It is important to use the Tab key to move to this field.) Type in the application's signature that was written down in step 2). Click in the box next to **Inited** to get an 'X' to appear in it.

5) Quit ResEdit, saving all changes.

The file that was created should have an icon that is the default document icon. It looks like a blank piece of paper with a bent corner (see Figure 7). If a user double clicks on the file's icon, the application should be launched. If the new file does not have both of these qualities, try the steps over again. It is important to note, however, that a few applications handle this extra file differently. For example, when launched with this file, MacWrite does not open an untitled document as it normally does. Other tricks need to be employed to handle applications which give errors when launched with the "calling" file. Look at Appendix II for an explanation on how to deal with these uncooperative applications.

**B. Making the "Calling" File's Icon Look Like the Application's**

This stage describes how to make the "calling" files appear on the desktop like the actual applications. Getting a general idea of how the Finder displays the icons on the desktop is very helpful.

The icon information for a volume is stored in an invisible file called the **Desktop** file. How the Finder determines which icon to display for a file, works in much the same way as how it determines what application to launch when someone double clicks on an icon. If the Finder is trying to display the icon for a file, it looks in **Desktop** for icon information associated with the creator of the file. For example, suppose that a file has a header as shown in Figure 5. The Finder looks in the **Desktop** file to see if there is icon information for the creator "MACA." If there is, then the Finder looks to see if this information contains the icon bit pattern for the file type "WORD." If the search is successful, then the Finder displays this icon, otherwise, the default icon for a file is displayed. Note that the Macintosh has two default icons: one for applications (file type "APPL") and one for documents (all other file types).

![Application Document Icons](image)

**Figure 7. The Two Macintosh Default Icons**

The icon information that is stored in the **Desktop** file is actually copied from the applications. When an application is written for the Macintosh, special resources are usually added to the application's resource fork so that the Finder can copy them. The resource fork is
one of the two parts of a Macintosh file (the other is the data fork). The resource fork contains formatted data called resources. Data such as fonts, menus, dialog boxes, and strings are stored in resources so that they may be easily changed without recompiling an application's code.

Icon information used by the Finder is copied from the application's resources named "ICN#", "FREF", and "BNDL". The "ICN#" resource contains the bit patterns of all of the icons, "FREF" contains a list which matches each file type with its icon, and "BNDL" completes the interface with the Finder by tying everything together. These three icon resources are copied to the Desktop file when an application is first moved onto the volume.

The icon resources can be examined using ResEdit. As an example, the window in the upper left hand corner of Figure 8 shows a few of the resources in MacWrite's resource fork. The other three windows display the contents of the resources "ICN#", "FREF", and "BNDL". Note that each resource type is broken down into ID numbers.

By double clicking on an icon in the "ICN#" resource, a window appears giving information about the icon, including the ID number associated with it. As an example, the window that appears by double clicking on the first icon is shown in Figure 9.

![Figure 8. Icon Resources in MacWrite's Resource Fork](image)
Figure 9. The Bit Pattern of the MacWrite Application Icon

By double clicking on a "FREF" resource, a window appears which tells what local icon id number and what file type are associated with the "FREF" ID number. The window shown in Figure 10 contains this information for the "FREF" ID number 128.

Figure 10. The Contents of the "FREF" resource, ID =128

After double clicking on a "BNDL" resource, a window appears similar to the one in Figure 11. The "BNDL" resource is used by the Finder to match the "FREF" ID numbers with the local file ID numbers and to match local icon ID numbers with "ICN#" ID numbers.

Page 11
<table>
<thead>
<tr>
<th>type</th>
<th>ICN#</th>
<th># of type</th>
<th>2</th>
</tr>
</thead>
<tbody>
<tr>
<td>localID</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rsrclD</td>
<td>128</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>type</th>
<th>FREF</th>
<th># of type</th>
<th>3</th>
</tr>
</thead>
<tbody>
<tr>
<td>localID</td>
<td>0</td>
<td></td>
<td></td>
</tr>
<tr>
<td>rsrclD</td>
<td>128</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Figure 11. The Contents of the "BNDL" Resource, ID Number 128

The diagram in Figure 12 graphically explains how each kind of ID number is used with each resource. The resources act as information tables and the ID numbers are used to index these tables.
To make the "calling" file's icon look like the application's, the "FREF" and "BNDL" resources need to be altered in such a way that the null file type of the "calling" file is recognized by the application. This is done by adding a "FREF" resource that identifies this file type and by updating the "BNDL" resource so that the Finder interface will be complete. The following procedure shows how to do this with MacWrite. This is followed by a general procedure for doing it with other applications.

Making MacWrite's "Calling" File's Icon Look Like the Application's

1) Make a copy of MacWrite and MacWrite.Call on a test disk (there should only be unimportant, erasable things on this disk). All of the following steps should be preformed on these copies only! Remember to use ResEdit cautiously.

2) Using ResEdit, open the MacWrite application. Open the "FREF" resource. Select New from the File menu. Enter the icon localID as shown in Figure 13 and then close the window.
3) Select the newly created "FREF" resource and then Get Info from the File menu. Change the number in the ID field to 132 and then close the window.

4) With this same resource selected, choose Open general from the File menu. Change the four 20's to four 00's and then close the window.

5) Close the "FREF" resource window and then open the "BNDL" resource. Select BNDL ID = 128 and then choose Open General from the File menu. Change the data so that it looks like Figure 15 and then close the window.

6) Open the BNDL ID = 128 in the normal fashion and then scroll to the end of the window. The last entry has zeros in both the localID and rsrcID fields. Change the zero in the rsrcID field to a 132.

7) Close all of the windows until the "Save before closing?" dialog box appears. Click Yes.

Figure 13. The Contents of the New MacWrite "FREF" Resource

Figure 15. How MacWrite's "BNDL" Resource Should be Changed.
8) Find the Desktop file on the test floppy and open it. Select the MACA resource. Choose Clear from the Edit menu. Close the window and click Yes when asked if you want to save before closing.

9) Quit ResEdit. The "Call" file should look like the application! If it does not, then erase the disk and try the procedure over again.

The procedure used for altering other applications is roughly the same as the one for MacWrite. If the "calling" file for a particular application works properly (i.e. no error occurs when the application is launched by it), then the following procedure describes how to properly alter the application. If, however, the "Calling" file does not work properly, then Appendix II should be consulted to see how and if the application can be protected with Application Calling.

General Procedure for Making the "Calling" File's Icon Look Like the Application's

1) Make a copy of the application and its '.Call' file on a test disk (there should only be unimportant, erasable things on this disk). All of the following steps should be preformed on these copies only! Remember to use ResEdit cautiously.

2) Using ResEdit, open the application which is to be used with Application Calling. Open the "FREF" resource. Open each of the "FREF" resources until the one with the information on file type "APPL" appears. Write down the number that is in the icon localID field (it is usually 0). Close the window.

3) The active window should now be the one showing all of the "FREF" resources. Write down the highest ID number shown. Select New from the File menu. Enter the ID number that was written down in step 2 in the icon localID field. Close the window.

4) Select the newly created "FREF" resource and then Get Info from the File menu. Change the number in the ID field to one plus the number written down in step 3 and then close the window.

5) With this same resource selected, choose Open general from the File menu. Change the four 20's to four 00's and then close the window.

6) Close the "FREF" resource window and then open the "BNDL" resource. Select BNDL ID = 128 and then choose Open General from the File menu. (If there is no "BNDL ID = 128" or if there is more than one "BNDL" resource, open the one who's owner is the application's signature. A "BNDL"'s owner can be determined by opening it in a normal fashion.)

7) Note that the numbers in the resource are in hexadecimal and run from left to right. Find the string of numbers 46 52 45 46. The two bytes following this string make up a two byte hexadecimal number. Increment it by one. For example, if the two bytes following the string were 00 0A, they should be replaced by 00 0B. Take this
new number, multiply it by four and write it down. In the example just given, the number written down would be \(44_{10} (000B \times 04 = 44_{10})\).

8) Skip the number of bytes written down in step 7 after the two byte number just incremented and then insert four zeros (00 00 00 00). For example, say the number was incremented to 00 03 in step 7. Twelve bytes should follow this number before the sequence 00 00 00 00 is entered. (It may turn out that this sequence of zeros is simply added at the end of the list of numbers in resource as in the case of MacWrite.) Close the window.

9) Open the same "BNDL" resource that was opened in step 6, but in the normal fashion and then scroll to the last entry of the "FREF" list. This entry has zeros in both the localID and rsrclD fields. Change the zero in the localID field to the number written down in step 2. Change the zero in the rsrclD field to one plus the number written down in step 3.

10) Close all of the windows until the "Save before closing?" dialog box appears. Click Yes.

11) Find the Desktop file on the test floppy and open it. Select the resource whose name is the application's signature. Choose Clear from the Edit menu. Close the window and click Yes when asked if you want to save before closing.

12) Quit ResEdit. The ".Call" file should look like the application! If it does not, then erase the disk and try the procedure again.

C. Moving the Applications to the Server

The next logical step after altering the applications is to move them and their "calling" files from the floppy disk to the server. This step can be extremely simple if the application to be protected has never been on the server before. In other words, if the Desktop file does not contain any information about a particular application, it is promptly updated when the altered application is copied onto the server. If, however, this application has already been copied onto the server (even if it were later removed), updating the Desktop file with the new information may involve forcing the Desktop file to be rebuilt.

Copy the altered applications and their "calling" files to the server. If the icons do not appear correctly, then the Desktop file should be forced to be updated with the resources stored in the altered applications. This can be accomplished by opening the Desktop file and removing those resources whose names are the signatures of the applications whose icons did not appear correctly. This is exactly what was done on the test floppy.

Once the icons for all of the files are showing up properly on the server, make a folder on the top directory called applications and another called rap. Next make a folder called l inside the rap folder. Put the ".Call" files in the applications folder and the real applications in the l
folder. The names of the ".Call" files may now be tailored to match your environment. For instance, the suffix ECN has been used for servers on the Engineering Computer Network at Purdue University. Once an appropriate name has been chosen for the "calling" file, the name of the real application should be changed so that it has the same name. This is because when an application is launched, the name of the real application appears at the top of the screen, and to maintain the standard Macintosh interface, the name of the "calling" file should appear at the top of the screen, not something else. The server is now ready to be arranged as outlined in Part I of this paper.

At a later date, it may be necessary to bring a newer version of a piece of software onto the server. Be sure to properly alter it before doing so.

D. Making the rap Folder Invisible

A bit in the rap folder's header information determines whether or not it is invisible. This bit is easily changed using ResEdit.

Procedure For Changing the Invisibility of the rap Folder
1) Launch ResEdit. Select the rap folder.
2) Choose Get Info from the File menu.
3) Click on the square beside the Invisible option (see Figure 16).

![Info for folder rap](image)

Figure 16. The Header for the rap Folder When It Is Invisible

4) Quit ResEdit, saving changes.
Summary

Application Calling can offer several valuable benefits: it can limit your disk space requirements and protect vendors from unauthorized copying of their software. Most importantly, it does this maintaining the standard Macintosh interface. In addition, it allows for quick and easy software updates for all of the users on the server.

The difficulty of using Application Calling is altering the applications and then updating the Desktop file with this information. This problem could easily be eliminated by the software companies if they released applications that already came with the "calling" file. This would only take them a few more minutes in software development time.
Appendix I. ResEdit

ResEdit is available to members only from:
Apple Programmer's and Developer's Association (APDA)
290 SW 43rd Street
Renton, WA 98055
membership fee: $20

It is also available as part of a Macintosh Programmer's Utilities kit:
Macintosh Development Utilities V.1.0
order # KMSDU1
price: $25.00

ResEdit is an application that can be used to edit or create Macintosh resources. When the
program is launched, a window for each mounted volume appears in which all of the folders and
files on that volume are listed.

Items are selected using the mouse and opened by double clicking. When some items are
opened, a window appears that has several fields. One can move from field to field by pressing
the Tab key or by using the mouse. The standard Macintosh editing techniques can be used
when changing text.

ResEdit gives the user an opportunity to change just about everything, and consequently,
an opportunity to ruin just about everything. Any outlined steps involving ResEdit should be
followed very carefully. If a mistake is made that seems too difficult to fix while editing
resources, quit ResEdit and do not save changes. This will restore the resources to their original
state.

For more information concerning ResEdit or resources consult Inside Macintosh.
Appendix II. Handling Uncooperative Applications

If an application does not respond properly when launched by the "calling" file, then another approach must be taken in order to use it with Application Calling. It is suggested that stage B. Making the "Calling" File's Icon Look Like the Application's be read before using this appendix.

The idea behind the method of handling uncooperative applications is to use a "calling" file whose file type is one that the application already recognizes. As an example, consider SuperPaint. Although it is not an uncooperative application, its "calling" file could be created using this alternate method. SuperPaint has a file called SuperPaint Prefs which saves a user's preferences. When someone double clicks on the icon of this file, SuperPaint is launched just as if someone had double clicked on the application's icon. Therefore, the SuperPaint Prefs file could be used as the "calling" file if its icon was changed to look like the application's.

The problem with this method is that the application originally used the "calling" file for some purpose and using it to launch the application normally prevents it from being used as before. SuperPaint will not run if it does not have a Prefs file in either the same folder as the real application or in the startup folder. This means that if this method is used, a Prefs file needs to be hidden in the same folder as the real application, inaccessible to the user. The Prefs file could not be kept in the startup folder because it looks like the application. The Prefs file that is being used as the "calling" file is indeed a Prefs file, but it would not make sense to require the user to keep it in the startup folder. How useful are applications that need to be launched from inside the startup folder?

A serious problem arises when an uncooperative application also does not have a file that behaves like a "calling" file. If this is the case, Application Calling cannot be used. It may be possible to write a small application that can call up the uncooperative one, but as of yet, we have been unsuccessful with such an endeavor.

The following steps describe the general procedure for making an uncooperative application's document look like the application. It assumes that the application has documents that behave just like "calling" files.

Changing an Existing Document's Icon to Look Like Its Application's

1) Make a copy of the application and the file whose icon is to be changed on a test disk (there should only be unimportant, erasable things on this disk). All of the following steps should be preformed on these copies only! Change the name of the file to the proper ".Call" name.

2) Using ResEdit, open the application to be protected. Open the "ICN#" resource.
Select the icon of the file which is to be changed and then double click on it (this would be the SuperPaint Prefs icon in the case of SuperPaint). Note that a box appears around an icon when it is selected. Write down the ID number shown in the window's title. Close the window. Select the same icon as before if it is not already. Choose Cut from the Edit menu.

3) Select the application's icon and choose Copy from the Edit menu. With the icon still selected, choose Get Info from the File menu. Change the number in the ID field to the number written down in step 2. Close the window.

4) Choose Paste from the Edit menu.

5) Close all of the windows until the "Save before closing?" dialog box appears. Click Yes.

6) Find the Desktop file on the test floppy and open it. Select the resource whose name is the application's signature. Choose Clear from the Edit menu. Close the window and click Yes when asked if you want to save before closing.

7) Quit ResEdit. The "Call" file should look like the application! If it does not, then erase the disk and try the procedure again.
Appendix III. Currently Altered Applications

The following is a list of applications which have been used with Application Calling by the School of Electrical Engineering at Purdue University as of November 1988. No applications have yet been encountered that could not be altered.

<table>
<thead>
<tr>
<th>Name</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>MacWrite 4.6</td>
<td>&quot;Calling file does not automatically open an untitled document. &quot;Calling file does not automatically open an untitled document.</td>
</tr>
<tr>
<td>MacWrite 5.0</td>
<td></td>
</tr>
<tr>
<td>SuperPaint 1.0</td>
<td></td>
</tr>
<tr>
<td>SuperPaint 1.1</td>
<td></td>
</tr>
<tr>
<td>LightSpeed Pascal 1.0</td>
<td></td>
</tr>
<tr>
<td>Microsoft Excel 1.5</td>
<td></td>
</tr>
<tr>
<td>LogicWorks 1.34</td>
<td></td>
</tr>
<tr>
<td>InfoUtil 1.01</td>
<td>Needed to correct a bug in this version's icon information.</td>
</tr>
<tr>
<td>InfoUtil 2.0</td>
<td></td>
</tr>
<tr>
<td>Device Librarian 1.32</td>
<td></td>
</tr>
<tr>
<td>MacSpin 2.0</td>
<td></td>
</tr>
<tr>
<td>Expressionist 2.0</td>
<td></td>
</tr>
<tr>
<td>MathType 1.53</td>
<td>The real application needed to be saved in a folder with read/write privileges, because the application writes to itself.</td>
</tr>
<tr>
<td>Cricket Graph 1.2</td>
<td></td>
</tr>
</tbody>
</table>
Appendix IV. Newer Versions of ResEdit

The procedures outlined in the text to create a 'calling' file and alter an application utilized a bug (in our case, a feature) of ResEdit. This bug was that a file could be created using the New command which had a file type of NULL characters. This special file type was not rejected by any applications altered by us and thus the method described in Appendix II for handling uncooperative applications did not need to be used.

Unfortunately for us, later versions of ResEdit no longer have this bug and the procedures given will not work. Therefore, an alternate procedure was drawn up which will work with these newer and 'better' versions of ResEdit. It should be pointed out that using a file type different from NULL characters is known to cause more uncooperative applications. However, unless you have a copy of ResEdit with this 'bug', you will have to use this less desirable solution.

A simple test to see if your copy of ResEdit has this bug, is to carefully follow the original procedures and to see if your 'calling' file looks like the application. If it does not, then you probably need to use the procedures outlined in this appendix.

General Procedure to Make the "calling" File for an Application:

1) Launch ResEdit. (See Appendix I for information concerning the use of ResEdit.) Find and select the application that is to be used with Application Calling. Choose Get Info from the File menu.

2) Write down the signature which is in the Creator field. (Case is important.) This signature is the application's creator type. Close the window.

3) Choose New from the File menu. In the New file name field, type in the name of the application plus a ".Call" suffix. For example, if the application is MacWrite, the name should be "MacWrite.Call". Click on OK. A window appears whose title is the name just entered. Close this window and the "calling" file is created.

4) If the file with the ".Call" suffix if is not already selected, do so. Get Info on this file. Move to the File Type field. Type in the four characters "CALL". Move to the Creator field. Type in the application's signature that was written down in step 2). Click in the box next to Init to get an 'X' to appear in it.

5) Quit ResEdit, saving all changes.
General Procedure for Making the "Calling" File's Icon Look Like the Application's

1) Make a copy of the application and its ".Call" file on a test disk (there should only be unimportant, erasable things on this disk). All of the following steps should be preformed on these copies only! Remember to use ResEdit cautiously.

2) Using ResEdit, open the application which is to be used with Application Calling. Open the "FREF" resource. Open each of the "FREF" resources until the one with the information on File Type "APPL" appears. Write down the number that is in the icon localID field (it is usually 0). Close the window.

3) The active window should now be the one showing all of the "FREF" resources. Write down the highest ID number shown. Select New from the File menu. Enter the ID number that was written down in step 2 in the icon localID field. Enter the four characters "CALL" in the File Type field and close the window.

4) Select the newly created "FREF" resource and then Get Info from the File menu. Change the number in the ID field to one plus the number written down in step 3 and then close the window.

5) Close the "FREF" resource window and then open the "BNDL" resource. Select BNDL ID = 128 and then choose Open General from the File menu. (If there is no "BNDL ID = 128" or if there is more than one "BNDL" resource, open the one who's owner is the application's signature. A "BNDL"'s owner can be determined by opening it in a normal fashion.)

6) Note that the numbers in the resource are in hexadecimal and run from left to right. Find the string of numbers 46 52 45 46. The two bytes following this string make up a two byte hexadecimal number. Increment it by one. For example, if the two bytes following the string were 00 0A, they should be replaced by 00 0B. Take this new number, multiply it by four and write it down. In the example just given, the number written down would be 4410 (000B x 04 = 4410).

7) Skip the number of bytes written down in step 6 after the two byte number just incremented and then insert four zeros (00 00 00 00). For example, say the number was incremented to 00 03 in step 6. Twelve bytes should follow this number before the sequence 00 00 00 00 is entered. (It may turn out that this sequence of zeros is simply added at the end of the list of numbers in resource as in the case of MacWrite.) Close the window.

8) Open the same "BNDL" resource that was opened in step 5, but in the normal fashion and then scroll to the last entry of the "FREF" list. This entry has zeros in both the localID and rsrclD fields. Change the zero in the localID field to the number written down in step 2. Change the zero in the rsrclD field to one plus the number written down in step 3.
9) Close all of the windows until the "Save before closing?" dialog box appears. Click Yes.

10) Find the Desktop file on the test floppy and open it. Select the resource whose name is the application's signature. Choose Clear from the Edit menu. Close the window and click Yes when asked if you want to save before closing.

11) Quit ResEdit. The ".Call" file should look like the application! If it does not, then erase the disk and try the procedure again.