Chapter 24
Standard File

Standard File has had extensive internal modifications to allow it to fully support GS/OS pathnames. All of calls work as they used to (except where noted below), but they are now more aware of GS/OS, and all the new calls that we have added support full GS/OS pathnames. Future applications should always try to use new calls as they fully support ALL types of GS/OS pathnames.

New Features

- Standard file allows a total of 13,107 files in a folder, and a maximum of 64K bytes for all file name strings in a folder.
  - Δ The 64K limit applies only to the size of the file name strings and does not include the size of the path string.
- Standard file now allows a maximum of 254 characters in a file name (new calls only)
- There is now a maximum of 508 characters allowed in a path name (new calls only)
- Standard file now uses Class 1 calls to fully support GS/OS.
- In order to make standard file code smaller, the List Manager is now used.
- Standard File now requires 4 pages of stack (3 for the list manager and 1 for everything else.)
- Similar to the Macintosh, Standard File will now leave the prefix set to the last selected path even if Cancel is pressed.
- On AppleShare volumes, Standard File will now scan the volume every 8 seconds for changes to the volume. If a change has occurred, the displayed list of files will be automatically re-built and re-displayed.
- Standard file now checks for write protected volumes and if write protected disables the SAVE & NEW FOLDER buttons in Put dialogs. This prevents Standard File from returning a file name which cannot be written.
- Standard file will now display a lock icon in the current path name on any disk which is write protected or not write enabled.
- Standard file now has calls to support multi selection Get calls.

Notes on the new Calls

When the new Standard File routines are called Prefix 8 is first checked to see if it contains a valid path. If so, then files in that path are displayed. If Prefix 8 is blank, then Prefix 0 is checked. If it contains a valid path, that path is first copied to Prefix 8 and then the files are displayed. Should Prefix 0 be blank, the next volume is checked. Anytime the path is changed during any standard file dialog (even if cancel is pressed), that path is placed back into Prefix 8. In addition, if the current path will fit into Prefix 0, it will be copied to there also. Old style Standard File calls always look into Prefix 0 for the starting path. It is then copied to Prefix 8 and all internal work is done using prefix 8. The results are placed back into prefix 0 and 8.

The original Standard File had two entries in its dialogs that are really no longer needed in the new tools. These were "Scroll" and "Files", who's jobs are now handled by the list manager. However, since old dialog templates must still be supported these items will remain in the standard template. Internally, however, a "copy" of the dialog header will be built into memory and these items will be removed (the numbers of the items will NOT be changed however) The "Scroll" item is useless and serves no function other than to hold a place in the template. The "Files" template DOES contain the bounding rect for the display box of file names. Any application which plans to supply its own dialog templates should set this item accordingly (see sample templates).

Variances from the old Standard File

- Errors are now returned by standard file (see each call for numbers). If more than 1 error should occur, only the first error encountered will be returned (e.g.: If the buffers passed in the reply record are BOTH too small, only path too small will be returned).

- In an effort to trap and define more internal errors, the new Standard File will now display a detailed information dialog on many internal errors and allow the process to be canceled (if an internal error occurred in the old standard file it simply aborted. This caused further calls to display the dialog in strange positions.). On fatal errors, the new Standard File aborts cleanly (if possible) and corrects the dialog positions before exiting.

- The new Standard File supports a wider range of key equivalents (listed below).

- Patching Dialog Templates: The new Standard File makes use of line edit by using StatText items in its dialog templates. The old version used UserItems. On entry to the new Standard File with a custom dialog template, the passed template will be "patched" to change the prompt entry item from UserItem to StatText. In addition, the program now uses a custom draw routine on the path entry item.. The passed dialog will be patched to contain a call to Standard File's custom draw routine (unless a routine is already provided in the template).

- Because of the preceding item, Standard File calls with custom dialog templates in ROM will not function.

- Although internally all Standard File calls (both Old and New) use Prefix 8 and therefor can have paths larger than 64 characters, Standard File will now display a dialog if entry to a folder would causes the path to be longer than 64 characters AND the call was an Old style call. In addition, or new calls Prefix 0 will shadow Prefix 8 only to the point where the path will continue to fit. At that point, Prefix 0 will contain the last legitimate path that would fit.

- The new Standard File will no longer return allow a SAVE or NEW FOLDER to be executed on a write protected disk.

- New calls have been added to support Multi-File selection. (For Get only).

Keyboard-Button equivalents:

The following keyboard-button equivalents are supported:

Key	Button "pressed"
ESC	CLOSE
Command up arrow	CLOSE
TAB	VOLUME
Command .	CANCEL
Command o	OPEN
Command O	OPEN
Command down arrow	OPEN
Command n	NEW FOLDER
Command N	NEW FOLDER

Verbs

Verbs used in Standard File are **Reference Verbs.** These are used for passing and returning information and indicate whether the data is referenced by a pointer, a handle, or a resource ID.

Reference Verbs

All standard file calls use the same verbs when a reference is passed. The following is a list of those verbs : a description of how they are used.

Verb	Name	Description
$0000	PointerVerb	The reference parameter points to a block of data. The length must be either fixed or somehow specified by the block of data.
$0001	HandleVerb	The reference parameter is a handle to a block of data.
$0002	ResourceVerb	The reference parameter is a resource ID which can be used to get the block of data.
$0003	NewHandle	Standard File will allocate new handle, size it correctly and pass it back to the calling program. The ID used to allocate the handle will be the one passed to Standard File startup. The calling program will be responsible for disposing of any handles created in this way.

Data Structures:

New class Reply Record

```
struct replyRecord
{
        word    good        TRUE for Open, FALSE for Cancel.
        word    type        File type.
        long    auxType     Auxiliary file information.
        word    nameVerb    verb to describe ref type (Pointer, Handle or NewHandle only).
                            This must be set to prior to calling!
        long    nameRef     Reference to a C1OutputStting
        word    pathVerb    verb to describe ref type (Pointer, Handle or NewHandle only).
                            This must be set to prior to calling!
        long    pathRef     Reference to a C1OutputStting
}
```

Valid verbs for the above Refs:

- 0000 The ref is a pointer to a C1OutputString.
- 0001 The ref is a handle to a C1OutputString
- 0003 The ref is undefined. The tool will create a new handle, size it correctly and return a handle to C1OutputString in the ref. This is the recommended verb.

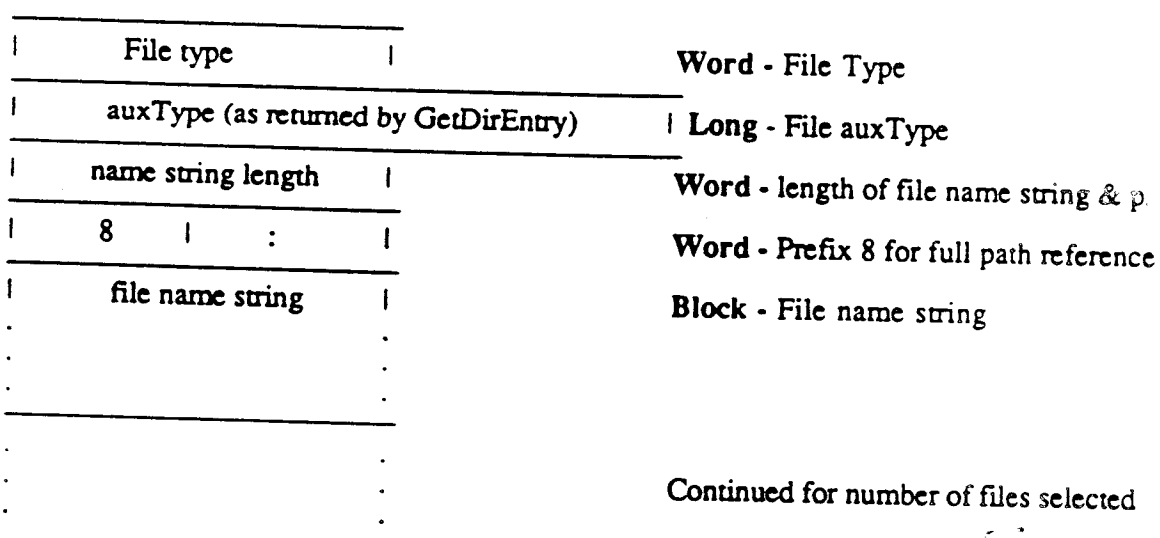
Note that for the first two verbs a 'Buffer Too Small' error ($004F) could occur if the resultant strings do not fit in to the buffer supplied. In the case of an error, as much of the string as possible will be put into the buffer and the error will be returned. The actual length of the string that SHOULD have been returned is saved in the string's length word. Also note that it is up to the application to release the handle supplied by verb 0003.

MultiFile class Reply Record

```
struct multireplyRecord
{
        word    good            # of files selected. FALSE for Cancel.
        long    namesHandle     Created handle to reply data record.
}
```

The reply record for MultiFile Get contains only two entries, 1). The number of files selected or FALSE if Cancel was pressed, and 2). á handle which is created by standard file with the calling program's ID which references the returned data record. The application is responsible for killing this handle when done.

The returned data record consists of entries each made up of a two bytes of file Type followed by 4 bytes of file Aux Type, followed by a length byte of the file name string, followed by the name string. The name string always starts with '8:', and prefix 8 is always left pointing to the path from which the files were selected (allows the returned string to be used as both a full path reference and an easily obtained file name reference), ie:



Standard File calls

SFAllCaps $0D17

inputs
- allCapsFlag WORD not used

outputs
- none

This call has been disabled and has no effect on any future Standard File Calls. However, applications may still make this call. It has been disabled to allow Standard File to follow the new standard of showing file names exactly as originally entered.

SFShowInvisible $1217

inputs
- space WORD Space for Old value of invisible state.
- InvisibleState WORD Display invisible files? 0=no 1=yes

outputs
- OldInviState WORD Previous value of invisible flag

This call is used to set a new state for displaying invisible files. When Standard File is started, invisible files will not be displayed and will not be passed to Filter Proc. routines. Passing true to this call will allow standard file to start showing invisible files.

SFReScan $137

inputs
- filterProcPtr Long Pointer to filter procedure; nil for no change.
- typelistPtr Long Pointer to typelist record; nil for no change.

outputs
- none

This call is used to allow an application to force the current list of files to be re-built and re-displayed. The application can elect to change the Type List, the Filter Procedure, both or neither. This will allow an application to easily display a new list of files of different type without exiting the tool.

The call is valid only while SFPGetFile or SFPGetFile2 are running and should only be called from within a Dialog Hook Procedure.

Errors: $1706 Bad call. SFPGetFile or SFPGetFIle2 is not running or the tool is inactive.

SFGetFile2 **$0E17**

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
filterProcPtr	Long	Pointer to filter procedure; nil for none.
typelistPtr	Long	Pointer to typelist record; nil for none.
replyPtr	Long	Pointer to a new class of reply record (see Data Structures).

outputs

none

This call displays the standard Open File dialog box and returns information about the file selected by the user. It uses GS/OS class 1 calls and allows selection of a file with a total path length of up to 763 characters (255 for file name and 508 for prefix). This call returns the file name in a new format which is more easily interfaced to GS/OS (and other tools).

Errors: $1701 Bad Reference Verb, prompt string.
$1704 Bad Reference Verb, ReplyRecord name
$1705 Bad Reference Verb, ReplyRecord path

more the about parameters
The *typelistPtr* points to a record containing a list of file types to display (set to nil to display all files):

numEntries class 1 type	**Word** - Total number of entries of the New style.
flags	**Word** - Flags describing action on next 6 bytes.
fileType 1	**Word** - First Class 1 file type (see flags).
auxType (as returned by GetDirEntry)	**Long** - First auxType (see flags)
.	.
.	.
last flags	**Word** - last flags describing action on next 6 bytes.
lastfileType	**Word** - last file type to display (see flags)
lastauxType	**Long** - last aux type (see flags)

Flags: The flag word describes the following actions:

Bit 15: Set to 1 to match all files with **this** fileType and **any** auxType.

Bit 14: Set to 1 to match all files with **this** auxType and **any** fileType.

Bit 13: Set to 1 display any files that match this fileType - auxType as specified by bits 14 & 15 but do not allow the user to select them (item is greyed out). Note that if you set this bit the file is NOT passed along to the filter proc. but simply displayed greyed out.

Bit 12 - 0: Reserved, must be 0.

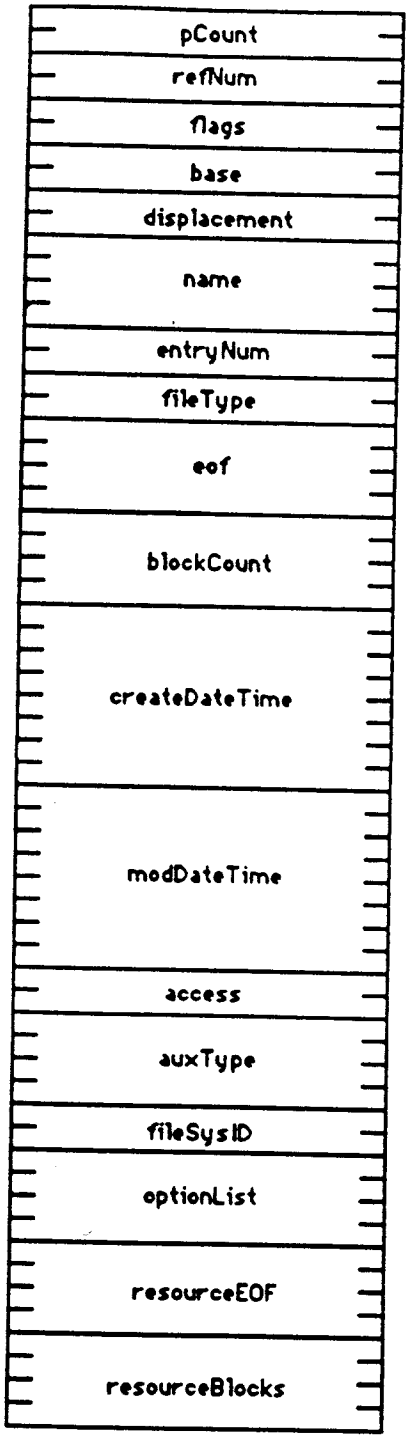
If further filtering is needed, the filter procedure pointed to by *filterProcPtr* also can be used to determine which files will be displayed. Similar to the original Standard File call, file entries are passed one at a time (on the stack) to the routine. Unlike the original Standard File, instead of a "File Directory entry" record pointer being passed, a pointer to a "GetDirEntry" record will be passed. See figure below or see the GS/OS reference, volume 1 on GetDirEntry for more information.

The procedure must strip off the 4 bytes of the directory entry off the stack and return with the results at the top of the stack. The results indicate what the procedure wants to do with the file (as shown below):

Value	Name	Description
0	noDisplay	Don't display file.
1	noSelect	Display the file but don't allow user to select it.
2	displaySelect	Display the file and allow it to be selected.

(do we need to add entries for read and write only directories?)

The GetDirEntry is described as follows (see preceding section on parameters):


pCount
refNum
flags
base
displacement
name
entryNum
fileType
eof
blockCount
createDateTime
modDateTime
access
auxType
fileSysID
optionList
resourceEOF
resourceBlocks

SFPGetFile2 $1017

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
filterProcPtr	Long	Pointer to filter procedure; nil for none.
typelistPtr	Long	Pointer to typelist record; nil for none.
dlgTempPtr	Long	Pointer to dialog template in memory.
dialogHookPtr	Long	Pointer to a routine called every time ModalDialog returns a hit.
replyPtr	Long	Pointer to a new class of reply record (see Data Structures).

outputs

none

This call displays the custom Open File dialog box and returns information about the file selected by the user. It uses GS/OS class 1 calls and allows selection of a file with a total path length of up to 763 characters (255 for file name and 508 for prefix). This call returns the file name in a new format which is more easily interfaced to GS/OS (and other tools).

See SFPutFile in original Standard File Documentation for a Description of dlgTempPtr and dialogHookPtr.

Errors:
$1701 Bad Reference Verb, prompt string.
$1704 Bad Reference Verb, ReplyRecord name
$1705 Bad Reference Verb, ReplyRecord path

$004F Buffer too small

SFMultiGet2 $1417

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
filterProcPtr	Long	Pointer to filter procedure; nil for none.
typelistPtr	Long	Pointer to typelist record; nil for none.
replyPtr	Long	Pointer to a multi-type reply record (see Data Structures).

outputs

none

This call displays the standard Open Multi-File dialog box and returns information about the file(s) selected by the user. It uses GS/OS class 1 calls and allows selection of a file names up to 253 characters long. File names, Type and Aux Type are returned via a handle which points to a record of files selected (see data structures). In addition, this call can also return folder names and the calling application should take care to check the file type before using the returned name(s). All file names are returned in the data structure as full paths by having the prefix number (8:) preceding the file name.

Notes on MultiFile Calls:

Multi-File calls include a new button labeled ACCEPT (please see dialog template notes). When in multi-file mode, both OPEN and ACCEPT will be enabled when a single file is selected. If the file is not a folder, either button will return the file name. If the selected item is a folder, OPEN will create a new list display showing the contents of the folder (as in regular Standard File). However, if ACCEPT is pressed, the folder name will be returned to the calling program as would any other file. OPEN is disabled when more than one item is selected and ACCEPT must be used to return the selected items. If the selected items contain a folder, the folder name will be returned to the calling program along with any other selected items in the returned data record. Double clicking is not allowed if more than one item is selected. Double clicking on a folder (as a single item) will OPEN the folder and display its contents. Double clicking on a file (as a single item) will return the file name as if ACCEPT was pressed.

Errors: $1701 Bad Reference Verb, prompt string.

SFPMultiGet2 $1517

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
filterProcPtr	Long	Pointer to filter procedure; nil for none.
typelistPtr	Long	Pointer to typelist record; nil for none.
dlgTempPtr	Long	Pointer to dialog template in memory.
dialogHookPtr	Long	Pointer to a routine called every time ModalDialog returns a hi
replyPtr	Long	Pointer to a new class of reply record (see Data Structures).

outputs

none

This call displays a custom Open Multi-File dialog box and returns information about the file(s) selected by the user. It uses GS/OS class 1 calls and allows selection of a file names up to 253 characters long. File names, Type and Aux Type are returned via a handle which points to a record of files selected (see data structures). In addition, this call can also return folder names and the calling application should take care to check the file type before using the returned name(s). All file names are returned in the data structure as full paths by having the prefix number (8:) preceding the file name.

See SFMultiGet2 for notes on Multi File calls.

See SFPutFile in original Standard File Documentation for a Description of dlgTempPtr and dialogHookPtr.

Errors: $1701 Bad Reference Verb, prompt string.
$1704 Bad Reference Verb, ReplyRecord name
$1705 Bad Reference Verb, ReplyRecord path

$004F Buffer too small

SFPutFile2 Call $0F17

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
origNameVerb	Word	verb which describes the following pointer.
origNameRef	Long	Reference to a C1 InputString holding the default file name
replyPtr	Long	Pointer to a new class of reply record (see Data Structures).

outputs

none

This calls displays the standard Save File dialog box and returns information about the file name entered by the user. It uses GS/OS class 1 calls and allows specification of a file with a total path length of up to 763 characters (255 for file name and 508 for prefix). This call returns the file name in a new format which is more easily interfaced to GS/OS (and other tools).

Errors:

$1701 Bad Reference Verb, prompt string.
$1702 Bad Reference Verb, name string.
$1704 Bad Reference Verb, ReplyRecord name
$1705 Bad Reference Verb, ReplyRecord path
$004F Buffer Too Small

Notes:

The original standard file call had a maxLen parameter which allowed the application to specify the maximum number of characters the user could type for a file name. This parameter has been removed from the new call as this limits the function of new FSTs. New FSTs may allow file names of *any* length and removal of this parameter prevents an application from summarily limiting the file name's length.

$1117 SFPPutFile2

inputs

whereX	Word	X coordinate of upper left corner of dialog box.
whereY	Word	Y coordinate of upper left corner of dialog box.
promptVerb	Word	verb which describes the following pointer
promptRef	Long	Reference to a Pstring as described by verb above.
origNameVerb	Word	verb which describes the following pointer.
origNameRef	Long	Reference to a C1 InputString holding the default file name
dlgTempPtr	Long	Pointer to dialog template in memory.
dialogHookPtr	Long	Pointer to a routine called every time ModalDialog returns a hit.
replyPtr	Long	Pointer to a class 2 reply record (see below).

outputs

none

This calls displays a custom Save File dialog box and returns information about the file name entered by the user. It uses GS/OS class 1 calls and allows specification of a file with a total path length of up to 763 characters (255 for file name and 508 for prefix). This call returns the file name in a new format which is more easily interfaced to GS/OS (and other tools).

See SFPPutFile in original Standard File Documentation for a Description of dlgTempPtr and dialogHookPtr.

Errors:

$1701	Bad Reference Verb, prompt string.
$1702	Bad Reference Verb, name string.
$1704	Bad Reference Verb, ReplyRecord name
$1705	Bad Reference Verb, ReplyRecord path
$004F	Buffer Too Small

Notes: The original standard file call had a maxLen parameter which allowed the application to specify the maximum number of characters the user could type for a file name. This parameter has been removed from the new call as this limits the function of new FST's. New FST's may allow file names of *any* length and removal of this parameter prevents an application from summarily limiting the file name's length.

Standard File Dialog Templates

The Standard File Tools Set allows one to provide custom dialog boxes for the Open File and Save File dialog boxes. To produce a custom dialog box, you use the SFPPutFile2, SFPGetFile2, or SFPMultiFile2 routines and provide a pointer to a dialog template in memory. A **dialog template** is a record passed to the Dialog Manager routine GetNewModalDialog. (see the tool call for how the dialog template is specified).

There are a few differences in the new Standard File Tool Set's dialogs. Several items from the dialog record are now used differently, but the overall size of the default dialog record has not changed. Old dialog templates should still work with the new calls but be aware that the tool may make patches to the dialog in memory in order to make it compatible with the new tool. Changes include: The bounding rectangle for the List Manager Display box is taken from the "Files" entry of each dialog template and copied to the internal List Manager Record. The number of files displayed is determined by subtracting 2 from the height of the rectangle and then dividing by 10. Thus a height of 82 would display 8 files. For best results it is suggested that the height be determined by reversing this formula and using an EXACT height (i.e.:(# of files * 10) + 2) in custom dialog templates or partial file names may be displayed. No other part of the "Files" entry is used. The "Scroll" entry is no longer used by non multi-file calls but is retained in the record as a place holder. **However**, to preserve space in the standard dialogs, the ACCEPT button record has been placed in the location previously left for the "Scroll" item (item 5 in GET). No change has taken place to the "Scroll" item in Put dialogs.

When Standard File calls are made, the passed dialog template header is copied to memory and items 5 & 7 (for GET), or items 6 & 8 (for PUT) are NOT copied into the new header. If a multi-file Get call is made, item 5 WILL be copied.

The following sections provide the templates that give the standard Open File and Save File dialog boxes. All of the templates depend on the following strings:

```
SaveStr     str   'Save'
OpenStr     str   'Open'
CloseStr    str   'Close'
DriveStr    str   'Volume'
CancelStr   str   'Cancel'
FolderStr   str   'New Folder'
AcceptStr   str   'Accept'
KbFreeStr   str   '^0 free of ^1 k.'  ;Dialog manager Routine replace ^0 & ^1 with
                                      ;real values from the disk.
PPromptStr  dc.b 'Save which file:'
PEndBuf     dc.b 0                    ;end of string byte

GPromptStr  dc.b 'Load which file:'
GEndBuf     dc.b 0                    ;end of string byte
```

The last two items are default strings for the prompt in Open file and Save file.

Templates for the standard Open File dialog box

For the Open File dialog box, the item part of the template must include the following items in this exact order:

Item	Item type	ID
Open button	buttonItem	1
Close button	buttonItem	2
Next button	buttonItem	3
Cancel button	buttonItem	4
Scroll bar	userItem+ItemDisable	5

* Note that the above item is no longer used by the new tool for non Multi-file calls but has been left in as place holder for compatibility with old dialog templates. For multi-file calls, item 5 becomes the new ACCEPT button.

Path	userItem	6
Files	userItem+ItemDisable	7

*Note that this item contains the bounding rect for the List Manager Record and serves no other purpose.

Prompt	userItem	8

640 mode

```
GetDialog640

     dc.w   0,0,114,400          ; drect of dialog (640 mode) recommended
     dc.w   -1
     dc.w   0,0                  ; reserved word
     dc.l   OpenBut640
     dc.l   CloseBut640
     dc.l   NextBut640
     dc.l   CancelBut640
     dc.l   Scroll640            ;dummy item or ACCEPT button
     dc.l   Path640
     dc.l   Files640
     dc.l   Prompt640
     dc.l   0

OpenBut640
     dc.w   1                    ;item #
     dc.w   53,265,65,375        ;drect
     dc.w   ButtonItem           ;type of item
     dc.l   OpenStr              ;Item descriptor
     dc.w   0,0                  ;Item value & bit flags
     dc.l   0                    ;color table ptr (nil is default)

CloseBut640
     dc.w   2                    ;item #
     dc.w   71,265,83,375        ;drect
     dc.w   ButtonItem           ;type of item
     dc.l   CloseStr             ;Item descriptor
     dc.w   0,0                  ;Item value & bit flags
     dc.l   0                    ;color table ptr (nil is default)

NextBut640
     dc.w   3                    ;item #
     dc.w   27,265,39,375        ;drect
     dc.w   ButtonItem           ;type of item
     dc.l   DriveStr             ;Item descriptor
     dc.w   0,0                  ;Item value & bit flags
     dc.l   0                    ;color table ptr (nil is default)
```

```
CancelBut640
     dc.w   4                        ;item #
     dc.w   97,265,109,375           ;drect
     dc.w   ButtonItem               ;type of item
     dc.l   CancelStr                ;Item descriptor
     dc.w   0,0                      ;Item value & bit flags
     dc.l   0                        ;color table ptr (nil is default)
```

```
Scroll640
     dc.w  5                          ;item # (DUMMY or ACCEPT button)
     dc.w  118,265,130,375            ;drect
     dc.w  ButtonItem                 ;type
     dc.l  AcceptStr                  ;Item descriptor
     dc.w  0,0                        ;Item value and bit flags
     dc.l  0                          ;color table

Path640
     dc.w  6                          ;item #
     dc.w  12,15,24,395               ;drect
     dc.w  UserItem                   ;type
     dc.l  PathDraw                   ;Item descriptor
     dc.w  0,0                        ;Item value and bit flags
     dc.l  0                          ;color table

Files640
     dc.w  7                          ;item #
     dc.w  25,18,107,215              ;Bounding rect for list manager display
     dc.w  UserItem+ItemDisable:type  ;does not matter
     dc.l  0                          ;does not matter
     dc.w  0,0                        ;does not matter
     dc.l  0                          ;does not matter

Prompt640
     dc.w  8                          ;item #
     dc.w  03,15,12,395               ;drect
     dc.w  LongStatText2+ItemDisable  ;type
     dc.l  GPromptStr                 ;Item descriptor
     dc.w  GEndBuf-GPromptStr         ;size of text
     dc.w  0                          ;Bit flags
     dc.l  0                          ;color table
```

320 mode

```
GetDialog320

     dc.w  0,0,114,260                ; drect of dialog (320 mode)
     dc.w  -1
     dc.w  0,0                        ; reserved word
     dc.l  OpenBut320
     dc.l  CloseBut320
     dc.l  NextBut320
     dc.l  CancelBut320
     dc.l  Scroll320                  ;dummy item or ACCEPT button
     dc.l  Path320
     dc.l  Files320
     dc.l  Prompt320
     dc.l  0

OpenBut320
     dc.w  1                          ;item #
     dc.w  53,160,65,255              ;drect
     dc.w  ButtonItem                 ;type of item
     dc.l  OpenStr                    ;Item descriptor
     dc.w  0,0                        ;Item value & bit flags
     dc.l  0                          ;color table ptr (nil is default)
```

```
CloseBut320
     dc.w  2                          ;item #
     dc.w  71,160,83,255              ;drect
     dc.w  ButtonItem                 ;type of item
     dc.l  CloseStr                   ;Item descriptor
     dc.w  0,0                        ;Item value & bit flags
     dc.l  0                          ;color table ptr (nil is default)

NextBut320
     dc.w  3                          ;item #
     dc.w  27,160,39,255              ;drect
     dc.w  ButtonItem                 ;type of item
     dc.l  DriveStr                   ;Item descriptor
     dc.w  0,0                        ;Item value & bit flags
     dc.l  0                          ;color table ptr (nil is default)

CancelBut320
     dc.w  4                          ;item #
     dc.w  97,160,109,255             ;drect
     dc.w  ButtonItem                 ;type of item
     dc.l  CancelStr                  ;Item descriptor
     dc.w  0,0                        ;Item value & bit flags
     dc.l  0                          ;color table ptr (nil is default)

Scroll320
     dc.w  5                          ;item # (Dummy Item or ACCEPT button)
     dc.w  118,160,109,255            ;drect
     dc.w  ButtonItem                 ;type
     dc.l  AcceptStr                  ;Item descriptor
     dc.w  0,0                        ;Item value and bit flags
     dc.l  0                          ;color table

Path320
     dc.w  6                          ;item #
     dc.w  14,6,26,256                ;drect
     dc.w  UserItem                   ;type
     dc.l  PathDraw                   ;Item descriptor
     dc.w  0,0                        ;Item value and bit flags
     dc.l  0                          ;color table

Files320
     dc.w  7                          ;item #
     dc.w  27,05,109,140              ;Bounding rect for list manager display
     dc.w  UserItem+ItemDisable       ;does not matter
     dc.l  0                          ;does not matter
     dc.w  0,0                        ;does not matter
     dc.l  0                          ;does not matter

Prompt320
     dc.w  8                          ;item #
     dc.w  03,05,12,255               ;drect
     dc.w  StatText+ItemDisable       ;type
     dc.l  GPromptStr                 ;Item descriptor = string
     dc.w  GEndBuf-GPromptStr         ;size of string
     dc.w  0                          ;Bit flags
     dc.l  0                          ;color table (0 = default)
```

Templates for the standard Save File dialog box

For the Save File dialog box, the item part of the template must include the following items in this exact order:

Item	Item type	ID
Save button	buttonItem	1
Open button	buttonItem	2
Close button	buttonItem	3
Next button	buttonItem	4
Cancel button	buttonItem	5
Scroll bar	userItem+ItemDisable	6
* Note that the above item has no function and is there as a place holder ONLY but must not be removed..		
Path	userItem	7
Files	userItem+ItemDisable	8
*Note that this item contains the bounding rect for the List Manager Record and serves no other purpose.		
Prompt	userItem	9
Filename	editItem	10
Free space	statText	11
Create button	buttonItem	12

640 mode

```
PutDialog640

     dc.w   0,0,120,320          ; drect of dialog (640 mode) recommended
     dc.w   -1
     dc.w   0,0                  ; reserved word
     dc.l   SaveButP640
     dc.l   OpenButP640
     dc.l   CloseButP640
     dc.l   NextButP640
     dc.l   CancelButP640
     dc.l   ScrollP640           ;DUMMY item
     dc.l   PathP640
     dc.l   FilesP640            ;contains bounding rect only
     dc.l   PromptP640
     dc.l   EditP640
     dc.l   StatTextP640
     dc.l   CreateButP640
     dc.l   0

SaveButP640
     dc.w   1                    ;item #
     dc.w   87,204,99,310        ;drect
     dc.w   ButtonItem           ;type of item
     dc.l   SaveStr              ;Item descriptor
     dc.w   0,0                  ;Item value & bit flags
     dc.l   0                    ;color table ptr (nil is default)
```

```
OpenButP640
     dc.w  2                    ;item #
     dc.w  49,204,61,310        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  OpenStr              ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

CloseButP640
     dc.w  3                    ;item #
     dc.w  64,204,76,310        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  CloseStr             ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

NextButP640
     dc.w  4                    ;item #
     dc.w  15,204,27,310        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  DriveStr             ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

CancelButP640
     dc.w  5                    ;item #
     dc.w  104,204,116,310      ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  CancelStr            ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

ScrollP640
     dc.w  6                    ;item # (Dummy item place holder only)
     dc.w  0,0,0,0              ;does not matter
     dc.w  UserItem             ;does not matter
     dc.l  0                    ;does not matter
     dc.w  0,0                  ;does not matter
     dc.l  0                    ;does not matter

PathP640
     dc.w  7                    ;item #
     dc.w  0,10,12,315          ;drect
     dc.w  UserItem             ;type
     dc.l  PathDraw             ;Item descriptor
     dc.w  0,0                  ;Item value and bit flags
     dc.l  0                    ;color table

FilesP640
     dc.w  8                    ;item #
     dc.w  26,10,88,170         ;Bounding rect for list manager display
     dc.w  UserItem+ItemDisable ;type
     dc.l  0                    ;Item descriptor
     dc.w  0,0                  ;Item value and bit flags
     dc.l  0                    ;color table
```

```
PromptP640
     dc.w  9                              ;item #
     dc.w  88,10,100,200                  ;drect
     dc.w  LongStatText2+ItemDisable      ;type
     dc.l  PPromptStr                     ;Item descriptor
     dc.w  PEndBuf-PPromptStr             ;size of text
     dc.w  0                              ;Bit flags
     dc.l  0                              ;color table

EditP640
     dc.w  10                             ;item #
     dc.w  100,10,118,194                 ;drect
     dc.w  EditLine+ItemDisable           ;type
     dc.l  0                              ;Item descriptor
     dc.w  32                             ;size of text
     dc.w  0                              ;Bit flags
     dc.l  0                              ;color table

StatTextP640
     dc.w  11                             ;item #
     dc.w  12,10,22,200                   ;drect
     dc.w  StatText+ItemDisable           ;type
     dc.l  KbFreeStr                      ;Item descriptor
     dc.w  0                              ;size of text
     dc.w  0                              ;Bit flags
     dc.l  0                              ;color table

CreateButP640
     dc.w  12                             ;item #
     dc.w  29,204,41,310                  ;drect
     dc.w  ButtonItem                     ;type
     dc.l  FolderStr                      ;Item descriptor
     dc.w  0                              ;size of text
     dc.w  0                              ;Bit flags
     dc.l  0                              ;color table
```

320 mode

```
PutDialog320

     dc.w  0,0,128,270          ; drect of dialog (320 mode)
     dc.w  -1
     dc.w  0,0                  ; reserved word
     dc.l  SaveButP320
     dc.l  OpenButP320
     dc.l  CloseButP320
     dc.l  NextButP320
     dc.l  CancelButP320
     dc.l  ScrollP320           ;DUMMY item
     dc.l  PathP320
     dc.l  FilesP320            ;Contains Bounding rect
     dc.l  PromptP320
     dc.l  EditP320
     dc.l  StatTextP320
     dc.l  CreateButP320
     dc.l  0

SaveButP320
     dc.w  1                    ;item #
     dc.w  93,165,105,265       ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  SaveStr              ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

OpenButP320
     dc.w  2                    ;item #
     dc.w  54,165,66,265        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  OpenStr              ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

CloseButP320
     dc.w  3                    ;item #
     dc.w  72,165,84,265        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  CloseStr             ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

NextButP320
     dc.w  4                    ;item #
     dc.w  15,165,27,265        ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  DriveStr             ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)

CancelButP320
     dc.w  5                    ;item #
     dc.w  111,165,123,265      ;drect
     dc.w  ButtonItem           ;type of item
     dc.l  CancelStr            ;Item descriptor
     dc.w  0,0                  ;Item value & bit flags
     dc.l  0                    ;color table ptr (nil is default)
```

```
Scroll1P320
     dc.w  6                        ;item # (Dummy item place holder only)
     dc.w  00,00,00,00              ;does not matter
     dc.w  UserItem                 ;does not matter
     dc.l  0                        ;does not matter
     dc.w  0,0                      ;does not matter
     dc.l  0                        ;does not matter

PathP320
     dc.w  7                        ;item #
     dc.w  00,10,12,265             ;drect
     dc.w  UserItem                 ;type
     dc.l  PathDraw                 ;Item descriptor
     dc.w  0,0                      ;Item value and bit flags
     dc.l  0                        ;color table

FilesP320
     dc.w  8                        ;item #
     dc.w  26,10,88,145             ;Bounding rect for list manager display
     dc.w  UserItem+ItemDisable     ;type
     dc.l  0                        ;Item descriptor
     dc.w  0,0                      ;Item value and bit flags
     dc.l  0                        ;color table

PromptP320
     dc.w  9                        ;item #
     dc.w  88,10,100,170            ;drect
     dc.w  StatText+ItemDisable     ;type
     dc.l  PPromptStr               ;Item descriptor
     dc.w  PEndBuf-PPromptStr
     dc.w  0                        ;Bit flags
     dc.l  0                        ;color table

EditP320
     dc.w  10                       ;item #
     dc.w  100,10,118,157           ;drect
     dc.w  EditLine+ItemDisable     ;type
     dc.l  0                        ;Item descriptor
     dc.w  32                       ;size of text
     dc.w  0                        ;Bit flags
     dc.l  0                        ;color table

StatTextP320
     dc.w  11                       ;item #
     dc.w  12,10,22,160             ;drect
     dc.w  StatText+ItemDisable     ;type
     dc.l  KbFreeStr                ;Item descriptor
     dc.w  0                        ;size of text
     dc.w  0                        ;Bit flags
     dc.l  0                        ;color table

CreateButP320
     dc.w  12                       ;item #
     dc.w  33,165,45,265            ;drect
     dc.w  ButtonItem               ;type
     dc.l  FolderStr                ;Item descriptor
     dc.w  0                        ;size of text
     dc.w  0                        ;Bit flags
     dc.l  0                        ;color table
```

Change History:

1/25/89	Added documentation for Lock Icon and lock disk changes.
1/26/89	Addition of Multi File documentation. (SFMultiGet2, SFPMultiGet2 and associated data structures)