HyperCard 2.1 Release Notes

May 6, 1991

Confidential - Apple Internal Use Only

I. Overview of this Release

HyperCard 2.1 is the version of HyperCard recommended for use with System 7.0. It is also compatible with System 6.0.5 and System 6.0.7.

II. Revisions for Support of System 7.0 Features

These revisions are grouped by their associated Toolbox Manager.

Alias Manager

When the Alias Manager is present, HyperCard 2.1 will call it to resolve alias files. Therefore, the following commands will work properly with alias files.

```
go to stack <alias file>
open <alias file> with <alias file>
open file <alias file>
write to file <alias file>
read from file <alias file>
picture <alias file>
```

The XCMD callback GetFilePath will also resolve alias files. Therefore, XCMDs that use this callback to locate files will require no revision to work with alias files. (The Picture XCMD is an example.)

In addition, if HyperCard finds an alias file when looking for the Home stack on startup, the alias file will be resolved.

AppleEvent Manager

When the AppleEvent Manager is present, HyperCard 2.1 will use it to process and send Apple events. HyperCard will recognize and handle the standard Apple events for opening documents, printing documents, quitting, executing a script, and evaluating an expression.

In addition, several of HyperTalk's built-in commands will have the added ability to send specific Apple events to other applications.

Apple event	Related HyperTalk command

oapp	open <application>
odoc	open <document> with <application>
pdoc	print <document> with <application>
quit	close <application>
clos	close <document> with <application>
dosc	send <expr> to program <programExpr>
eval	request <expr> of program <programExpr>

If you want to launch another application or open a document with another application, use the HyperTalk command "open". If you want to print a document with another application, use the command "print". If you want to close a document in another application or quit another application, use the command "close". If you want to execute a script or macro in another application, use the HyperTalk command "send". If you want to evaluate an expression in another application, use the command "request". Please note that of all possible Apple events, only 'oapp', 'odoc', 'pdoc', and 'quit' are universally supported. All other Apple events that HyperCard can send will be effective only if the target application includes specific support for them. For example,

```
close "My Expense Report" with "FabCalc"
```

will work properly only if the application FabCalc supports the 'clos' Apple event.

The "open", "print", and "close" commands will work only with applications that reside on the same machine as HyperCard. The "send" and "request" commands will work with any linkable program running on the network. See "Determining the Target Program", below.

Sending scripts to other programs

The send command in HyperTalk will send a "do script" Apple event from HyperCard to another application running remotely. It can be used to send a script to any program that understands the standard 'dosc' Apple event. By default, HyperCard waits for a reply from the target program before continuing. However, you can specify that you don't want to wait for a reply.

Examples:

```
send "make waves" to program "De Anza 6/2nd:WildCraft:HyperCard"
     send "build {project}" to program "MPW Shell" without reply
```

Evaluating expressions in other programs

The request command in HyperTalk will send an "evaluate expression" Apple event from HyperCard to another application running remotely. It can be used to send an expression to any program that understands the standard 'eval' Apple event. The value of the expression will be put into the local variable "it".

Examples:

```
request "the name of this stack" of program "HyperCard"
     request "{target}" from program "MPW Shell"
```

Handling failures

The "send", "open", "print", "close", and "request" commands set "the result" as follows when they fail.

Condition	the result
Target program returned error string in reply.	the string
Target program timed out.	"Timeout"
Target program didn't handle event.	"Not handled by target program"
Target program returned error number in reply, or AESend returned some other error.	"Got error <errorNum> when sending Apple® event."
User canceled the "Link to program" dialog.	"Cancel"

When sending Apple events to another program, if HyperCard has not established a link with the target program, the user will be presented with a dialog, through which the link will be established (Figure 1). If a link has already been established between HyperCard and the target program, the Apple event will be sent without further user interaction.

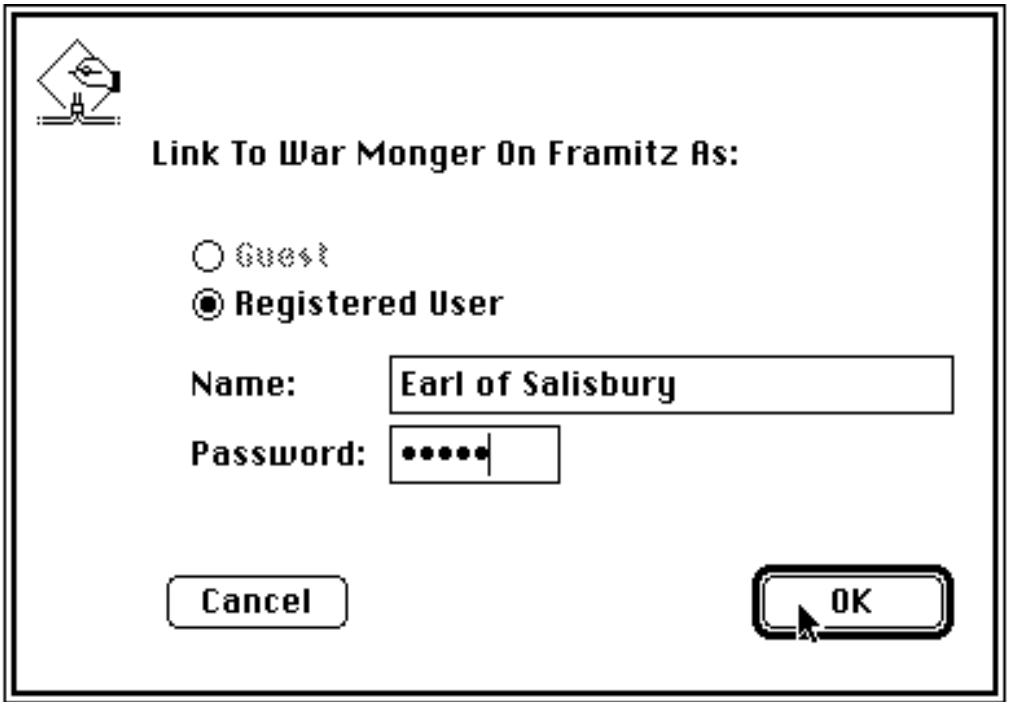

Figure 1

Determining the Target Program

The target program for Apple events will be specified by a string in the following format:

```
zone:machine:program
```

Examples:

```
"Fifth Floor:Gizmo:MacWrite"
"Baby mac:MacWrite"
      -- zone omitted; HyperCard assumes same zone
MacWrite
      -- zone and machine omitted; HyperCard assumes same zone,
            same machine
```

HyperCard's address on the network is contained in the property, "the address".

You can use the itemDelimiter property to parse addresses. For example, given a string that specifies a target program, the following HyperTalk function will return the name of the program if you send it a colon as the delimiter and the string as the text.

```
function lastHCItem delim,theText
  put the itemDelimiter into savedDelimiter
  set the itemDelimiter to delim
```

```
put the last item of theText into lastItem
set the itemDelimiter to savedDelimiter
return lastItem
end lastHCItem
```

Choosing a target program

The answer command will be extended for use with the Program Linking dialog, so that scripters can allow users to select a program to link to.

```
answer program prompt { of type <factorList> }
```

When used in this way, the answer command displays the PPC browser, from which the user can select any program running on any machine connected to the AppleTalk network (Figure 2). A string representing the program the user selects is placed into the local variable "it".

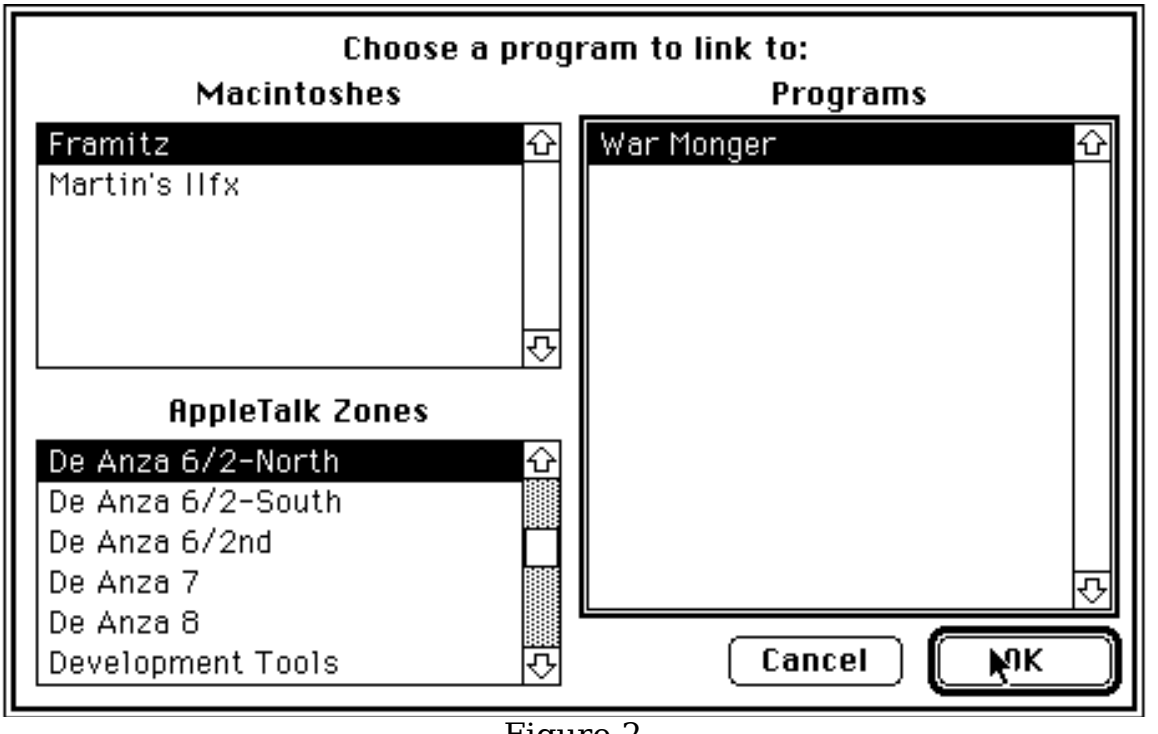

Figure 2

Examples:

```
answer program "Where did you say that program was again?"
answer program "Choose a spelling checker:" of type "Spellcheck"
```

In addition, a scripter can determine which programs are running on the same machine as HyperCard by examining the new HyperTalk function, "the programs". It returns a return-delimited list of all the linkable programs currently running on the same machine as

HyperCard.

Handling Apple events

When HyperCard receives an Apple event, it will send a new HyperTalk system message, "appleEvent", to the current card, along with parameters that enable a script to determine the class, id, and the sender of the Apple event. For example, when the Finder sends HyperCard an Apple event of class 'aevt' and id 'odoc', HyperCard sends the HyperTalk message "appleEvent aevt,odoc,Finder" to the current card.

A script that handles an appleEvent message can gather the parameters of the Apple event by using the new HyperTalk command "request".

```
request appleEvent data { [ of | with ] keyword <expr> }
```

This command puts the parameter or attribute with the specified keyword into the local variable "it". For example, you can obtain a parameter of keyword "errs", the standard Apple event keyword for an error string, as follows:

```
request appleEvent data with keyword "errs"
put it into errorString
```

If there is no attribute or parameter with the keyword you specify, HyperCard sets the result to "Not found".

If you don't supply a keyword, HyperCard assumes you're requesting the direct object of the Apple event, which is defined by the Apple event manager as the parameter with keyword "----". The request command also supports several special cases for important attributes of Apple events:

```
request appleEvent class
request appleEvent id
request appleEvent sender
request appleEvent return id
```

HyperCard will attempt to convert all the Apple event parameters to strings, for use in HyperTalk. The Apple event manager automatically handles the conversion of numerical forms to strings; HyperCard installs coercion handlers for alias records, return ids, process serial numbers, target ids, class and id types, and lists.
If an incoming Apple event specifies that user action is not permissible, the global property lockErrorDialogs will automatically be set to TRUE (see below). The user will be able to override this setting.

Example: a scripter defines an event of class 'WILD' and id 'cnvt' as a request to convert a 1.2.5 stack to the 2.0 file format. The direct object of the Apple event is defined to be the pathname of the stack. The following script will do the trick.

```
on appleEvent eventClass,eventID,sender
   if eventClass & eventID is not "WILDcnvt"
   then pass appleEvent
   else
     request appleEvent data
      if it is empty then exit appleEvent
      go to stack it
      if the result is not empty then exit appleEvent

      if there is a menuItem "Convert Stack..." in menu "File"
      then doMenu "Convert Stack..." without dialog
  end if
end appleEvent
```

If HyperCard receives the appleEvent message, either because it was not intercepted by a script or because it was passed to it by a script, HyperCard will check whether the current Apple event is of the types it knows how to respond to, and if so, will treat it appropriately. HyperCard 2.1 will recognize the following standard Apple events: 'odoc', 'pdoc', 'quit', 'dosc', and 'eval'.

HyperCard responds to the 'odoc' event by checking whether the specified files are stacks and, if so, opening them, each in a new window.

Event ID: 'odoc'

Event parameters:

keyDirectObject: a list of alias records that refer to HyperCard stacks.

HyperCard's response:

The stacks are opened, each in a new window.

HyperCard responds to the 'pdoc' event by checking whether the specified files are stacks and, if so, opening them one at a time in a new window, printing them, and closing them again.

Event ID: 'pdoc'
Event parameters:

keyDirectObject: a list of alias records that refer to HyperCard stacks.

HyperCard's response:

The stacks are opened, one at a time, printed, and then closed.

HyperCard responds to the 'quit' event just as it would if "Quit HyperCard" were chosen from the File menu.

Event ID: 'quit'

Event parameters: none required

HyperCard's response:

HyperCard first permits external windows, such as the Script Editor, to clean up, and if they are ready to quit, HyperCard cleans up and exits to the Finder.

HyperCard responds to a 'dosc' event by compiling and executing the HyperTalk statements sent to it as the direct object of the event.

Event ID: 'dosc'

Event parameters:

keyDirectObject: a string containing return-delimited HyperTalk statements.

HyperCard's response:

The statements are executed.

HyperCard responds to an 'eval' event by evaluating the expression sent to it as the direct object of the event and putting the value into the direct object of the reply.

Replying to Apple events

Scripters will be able to reply to Apple events from HyperTalk with a new command, "reply".

```
reply <expr> { with keyword <expr> }
```

During a chain of execution that begins with the HyperTalk system message "appleEvent", the reply command will add parameters to the default reply that will be returned by the AppleEvent Manager to the program that sent the original Apple event to HyperCard. If you don't specify a keyword for the reply parameter, the parameter will become the direct object of the reply. If you wish to return an error string, you can use the following form of the reply command:

```
reply error <expr>
```

This is equivalent to:

```
reply <expr> with keyword "errs"
```

The reply event is sent when the chain of execution is complete.

For example, the following script will handle Apple events of class 'WILD' and type 'defn' by searching for a string in a background field named "Glossary Entry" and returning the contents of a background field named "Definition".

```
on appleEvent eventClass,eventID,sender
   if eventClass is "WILD" and eventID is "defn" then
     request appleEvent data
      find it in field "Glossary Entry"
      if the result is empty  -- successful find
      then reply field "Definition"
      else reply error "Not found"
  else pass appleEvent
end appleEvent
```

The "request" and "reply" commands set the result to "No current Apple® event." when there is no current Apple event to handle.

III. Miscellaneous Revisions

Menus

When you ask for the names of the Apple, Help, and Application menus from HyperTalk, you get "Apple", "System Help", and "Application", respectively.

Error Handling

In order to execute scripts without interaction with the user, as well as to provide a better means for testing HyperCard's error messages, it will be possible to lock error dialogs and to put HyperTalk into a "quiet mode" for script execution. When HyperTalk encounters an error while in "quiet mode", it aborts execution of pending scripts just as it normally would, but instead of displaying a dialog with an error message, it sends a "errorDialog" message to the current card with the error message as its parameter. This message substitutes for the first "idle" message that would be sent in normal mode after HyperTalk cleans up and HyperCard returns to its main event loop.

Example:

```
set lockErrorDialogs to TRUE  -- puts HyperTalk into "quiet mode"
lock error dialogs  -- an alternate form
unlock error dialogs
```

When HyperCard receives an Apple event for which the sender has specified that no user interaction is allowed, it will automatically set lockErrorDialogs to TRUE before handling the event.

Determining which version of System Software is running

A new function will be added to HyperTalk, "the systemVersion". It returns the version of system software is running in a decimal string. For example, it will return 6.07 instead of 6.0.7, to allow scripters to use it with HyperTalk's arithmetic operators.

Specifying Windows

You can now refer to windows by number and by id as well as by name. Therefore, the following commands are now valid.

```
get the id of the card window
set the loc of window 1 to 50,50
hide last window
```

Both the Picture and Palette XCMDs now send the id of a newly created window as an additional parameter to its open message. For example, after the Picture XCMD creates a new window, it sends the following message to the current card:

```
openPicture <name of window>,<id of window>
```

Read and Write

The HyperTalk "read" and "write" commands have been enhanced. The following examples are now valid.

```
read from file "Fred" at 100 for 12
read from file "Fred" until end  -- 'eof' works too
write myVar to file "Fred" at 200
write moreStuff to file "Fred" at end  -- 'eof' works too
```

In addition, the 16K limit on the amount of text that could be read at once has been removed. If the amount of text you ask for won't fit in memory, HyperCard sets the result to "Not enough memory to read from file."

Getting that intransigent internal modem for the Portable to work with "dial"

A new global property, "the dialingTime", will be added, especially for users of the internal modem for the Macintosh Portable. It determines how long HyperCard waits, in ticks, after sending the dial string, before closing the serial connection with the modem. The default is 180 (3 seconds).

Owners

Windows and cards now have an "owner" property. For windows, the owner is determined as follows.

Creator	Owner
HyperCard	"HyperCard"
desk accessory or driver	"System"
XCMD	name of XCMD
anything else	"Unknown"

The owner of a card is the name of its background.

More Parameters Available

The parameters for the "start", "stop", and "set" commands are now available within overriding handlers.

IV. Summary

New messages: appleEvent, errorDialog
New commands: request, reply
New functions: programs(), systemVersion()
New properties: the address, the itemDelimiter, the lockErrorDialogs, the dialingTime, the owner
New constants: comma, colon

Enhanced commands: open, print, close, send, read, write, answer, lock, unlock, set, start, stop