AppleScript

Err:=AppleScript(Commands;Result)

Runs a script using AppleScript. The AppleScript editor doesn't need to be running. The command will be executed by the default system scripting language, which will usually be AppleScript, although other languages such as Frontier™ can be installed as a scripting component (see the AppleScript developer's kit for more information). The result will be 0 if the script runs successfully, -1 if AppleScript isn't installed, -2 if no scripting language is available, or -1753 if the script has a syntax error (other errors may be possible depending on the scripting language). You can explicitly choose a language other than AppleScript with the SetComponent command.

Example:

```
$q:=char(34) ` double-quote
$cmd:="display dialog "+$q+TextVar+$q
$err:=AppleScript($cmd;$result)
```

DoScript

Err:=DoScript(Target;Command)

Sends a "do script" AppleEvent to an application which must already be running. 'Command' is a text variable, field, or literal and will be in a format specific to the receiving application. For HyperCard®, this should be any HyperTalk command. For Frontier™, it should be a UserTalk command. Other applications will have different command formats. This can also be used with Excel® 4.0 to execute any function or macro. NOTE: DoScript won't wait for the commands to complete, so if you need to synchronize operations in another application, you should use SendWithReply instead.

Example:

```
 if (IsRunning("WILD")=0)

$err := Launch("WILD";"")
 end if
 $err := MakeAddress("WILD";$ad)
 if ($err = 0)

$err:=DoScript($ad;"Go To Stack 'my stack'")


$err:=DoScript($ad;"Go To Next Card")


$err:=DoScript($ad;"Print Card")


$err := DisposeAddress($ad)
 end if
```

Evaluate

Err:=Evaluate(Target;Command;Result)

Sends an "evaluate" message to an application and waits for the result, which will be of type TEXT. This is most often used with HyperCard® and SuperCard®.

Example:

```
$err:=MakeAddress("WILD";$hc)
if ($err = 0)
   $err:=Evaluate($hc;"the long name of the target";$result)
   alert($result)
   $err:=DisposeAddress($hc)
end if
```

Frontier

Err:=Frontier(Command;Result)

Sends a command to UserLand Frontier™ (which must already be running on your machine) and returns the result or an error message. If you pass the name of a Frontier™ object (such as User.Name) rather than an executable command, the contents of that object will be returned.

Example:

```
$err:=Frontier("User.Name";$myName)
If ($err=0)
   $q:=Char(34)
   $err:=Frontier("msg("+$q+"Hello, "+$myname+$q+")";$rep)
   $err:=Frontier("Frontier.BringToFront()";$rep)
   $err:=Frontier("edit(readme)";$rep)
End if
```

QuicKeys

Err:=QuicKeys(Macro Name)

Executes a QuicKeys™ sequence. You must be running QuicKeys2 v2.1 or later with CEIAC. The sequence must be installed in the Universal keyset or 4th DIMENSION®'s keyset. This command will also work with QuicKeys 3.0 and QuicKeys toolbox.

GetComponent

Err:=GetComponent(code)

Returns the 4-character ID of the default scripting language. For AppleScript, it will return “ascr”.

SetComponent

Err:=SetComponent(code)

Changes the default scripting language component. To use AppleScript, specify “ascr”. For QuicKeys® 3.0 use “QKCL”. For Frontier™, use “LAND”. You can use the command ListComponents to obtain the list of all scripting language codes. NOTE: QuicKeys Script is not compatible with 4D 3.0.5; you must use the QuicKeys command instead to execute sequences by name. With 4D 2.2.3 you can use the AppleScript command to run QuicKeys 3.0 scripts.

ListComponents

Err:=ListComponents(type;count;codes;names)

Returns a list of all components of the specified type. ‘Type’ should be a 4-character component type code. If you pass a null string, it will return a list of scripting language components. ‘Count’ should be a long integer variable, which will receive the number of components available. ‘Codes’ should be a string array which will receive a list of 4-character component IDs. ‘Names’ should be a text array which will receive the component names. If the arrays don’t exist, they will be created automatically.

Example:

```
Err:=ListComponents("osa ";howmany;codes;names)
Err:=GetComponent(theCode)
Which:=FindInArray(codes;theCode)
Alert("Current scripting language is:"+nameswhich)
```