AllowAccess

AllowAccess(Flag)

Turns on or off System 7 Pack™'s standard AppleEvent handling (built-in handlers for DoScript and data & structure-access events). The event handling is initially turned off. Calling this function with a non-zero Flag will enable standard events. Calling this with 'Flag' set to zero will disable standard event handling. If an application tries to send an AppleEvent when they're disabled, it will get back an error message 'AppleEvents not allowed at this time'. This doesn't affect handlers installed with HandleAEVT.

GetAEMessage

GetAEMessage(Msg)

Gets the text-based direct object received with the last AppleEvent. This will usually be called in a procedure installed via HandleAEVT. To make sure that what you received is text, you should call GetAEType and make sure that the type isn’t PICT. Aany numeric type can be converted to text and will be returned as a string by GetAEMessage, but pictures will result in a null string. You can also use the low-level interface to extract multiple pieces of data with finer control.

GetAEPict

Err:=GetAEPict(aPicture)

This should be called in a procedure installed via HandleAEVT to retrieve picture data from the AppleEvent. Use this function to handle an event that was sent with AESendPict.

Example:

```
HandleAEVT(“DEMO”;”PICT”;”Receive Picture”)

-- Receive Picture --
GetAEType($type)
If ($type # “PICT”)

Alert(“Wrong data type received:”+$type)
Else

$err:=GetAEPicture(picVariable)

if ($err#0)

    Alert(“Error receiving picture:”+string($err))

end if

Create Record([Pictures])

[Pictures]thePicture:=picVariable
```

```
Save Record([Pictures])
End if
```

GetAEType

GetAEType(theType)

Returns the data type of the direct object of the last AppleEvent. This should be called in a procedure installed via HandleAEVT to ensure that the received data is in the expected format. A procedure that expects PICT will be able to receive only PICT by calling GetAEPict.   A procedure that expects text can receive any text or numeric type by calling GetAEMessage.

GetReturnAddr

GetReturnAddr(Target)

Gets the return address of the last AppleEvent received. The target is identical to one returned by SelectAddress and may be used in any of the AppleEvent functions which take a target address. It should be disposed of when you finish with it. This should be used in a procedure installed via HandleAEVT.

HandleAEVT

Err:=HandleAEVT(Class;ID;Command)

Installs a 4D® command line or procedure to be executed in response to a particular AppleEvent. Don't try to replace any of the 4 required events (class 'aevt' & ID 'odoc', 'oapp', 'pdoc', or 'quit') or one of the standard events handled by System 7 Pack™.

NOTE: versions 3.1 and later of S7P allow you to remove the built-in ‘aevt’,’quit’ handler, after which you can add your own quit procedure which can do any necessary clean up and then call QUIT 4D.

4D 3.0 NOTE: All AppleEvent handlers execute in a process called Apple Event Manager. This process has no window associated with it. Therefore, you must never call MESSAGE without first creating a window.

Example:

```
err:=HandleAEVT("DEMO";"TEST";"DemoHandler")

-- DemoHandler --
GetAEMessage($txt)
GetReturnAddr($sender)
$err:=AESend($sender;"DEMO";"ACK!";$txt)
Alert($txt)
```

```
$err:=DisposeAddress($sender)
```

IgnoreAEVT

Err:=IgnoreAEVT(Class;ID)

Removes a previously installed AppleEvent handler.   Note: you may use this command to remove the built-in handler for ‘aevt’,’quit’ to allow a custom quit handler to be used. Don't use this to remove any of 4D's standard event handlers other than ‘quit’.

ProcessAEVT

Allows AppleEvents to be received and processed while 4D is in a tight processing loop. You only need to use this function if a loop is waiting for an AppleEvent to set some flag before it exits, since 4D normally doesn’t poll for events during such loops. THIS COMMAND SHOULD NEVER BE NECESSARY IN 4D V3.0 or later.

Example:

```
MyEventFlag := False

` will be set in AE handler
while (MyEventFlag = False)

ProcessAEVT
end while
```