Technical Note IC505

Apple Event Manager Q&As

CONTENTS

This Technical Note contains a collection of archived Q&As relating to a specific topic--questions sent the Developer Support Center (DSC) along with answers from the DSC engineers. Current Q&As can be found on the Macintosh Technical Q&As web site.

[Oct 01 1992]

SIZE resource isHighLevelEventAware bit and high-level events

Date Written: 11/17/92

Last reviewed: 7/2/93

When I try to send an OpenSelection Apple event to the Finder, I get a -903 error after calling AESend. Could you tell me what causes this error and how I can overcome it?

———

The application is returning the -903 error because it isn't completely set up to send and receive Apple events. This is why you're getting the error only after calling AESend. The problem is that the isHighLevelEventAware bit isn't set in your SIZE resource, meaning that your application doesn't send or receive high-level events. For more information on setting the isHighLevelEventAware bit, see *Inside Macintosh: Macintosh Toolbox Essentials,* pages 2-115 through 2-119 (or *Inside Macintosh* Volume VI, pages 5-14 through 5-17).

AETracker tracks Apple events

Date Written: 1/22/93

Last reviewed: 4/1/93

Do you have any software that detects Apple events? I need something that will let me know when my application sends Apple events and if the receiving application processes them correctly.

———

If brevity is the soul of wit, then I'm one funny dude: Try AETracker on the Developer CD (in the Tools & Apps: OS/Toolbox: Apple Events folder).

Descriptor disposal after AEPutXXXX

Date Written: 12/7/92

Last reviewed: 6/14/93

When a request for information is passed to me through an Apple event, the direct object parameter of my reply event is a descriptor list that includes an AERecord of my information. When I use AEPutPtr and the AEPutParamDesc, is the data copied or merely referenced? Should I be disposing of the AERecord and/or the descriptor list, or should I expect AEProcessAppleEvent to dispose of them?

———

Whenever you make an AEPutXXXX call, the Apple Event Manager *copies* the data you put into the list, event, or record, so as soon as you do an AEPutXXXX you can dispose of the data you put. Thus, the following is correct:

```
AEPutParamDesc(&theEvent, keyDirectObject, &theSpec);
AEDisposeDesc(&theSpec);
```

And so is this:

```
HLock(myTextHandle);
AEPutParamPtr(&theEvent, keyDirectObject, typeText,
  (Ptr)*myTextHandle, GetHandleSize(myTextHandle));
DisposeHandle(myTextHandle);
```

The *only* two descriptors disposed of by the Apple Event Manager itself (at the conclusion of AEProcessAppleEvent) are the original Apple event and the reply Apple event. So, anything that you create and manipulate yourself should be disposed of by you when you add it to another Apple event record or when you're done with it. The only two you don't dispose of yourself are theEvent and reply, which are passed to you, as in:

```
pascal OSErr AEXXXHandler(AppleEvent *theEvent, AppleEvent *reply,
  long refIn)
```

This even holds true for AESend. When you send an event, you can immediately dispose of your copy of the event, as follows:

```
AESend(&myEvent, nil, kAENoReply, kAENormalPriority,
  kAEDefaultTimeout, nil, nil);
AEDisposeDesc(&myEvent);
```

Limit for number of files opened from the Finder using 'odoc' event

Date Written: 11/9/92

Last reviewed: 6/14/93

Is there some limitation on the number of files you can open from the Finder using the 'odoc' event? I wrote a simple drag-and-drop application, using the 'odoc' event. When I tried to drag and drop 190 files over it, it froze.

———

These limits aren't officially documented anywhere. For drag and drop, the limit should be whatever causes the event to be passed going over 64K, since the High-Level Event Manager doesn't handle anything over 64K right now. 600 files at roughly 250 bytes per alias would push you way over the limit. At that point the Finder is probably not able to recover gracefully from a HLE error.

Construction of Apple events

Date Written: 4/9/91

Last reviewed: 6/14/93

How do I pass an Apple event to an application at launch time?

———

You can coerce the Apple event to a data type that doesn't need to be pre-addressed.A sample called ProcDoggie on the Developer CD demonstrates this. If you look in the UProcessUtils.inc1.p file, you'll find this comment:

"To create the kAEOpenDocuments or kAEPrintDocuments Apple event, a target address descriptor is needed. Because we're converting the Apple event into a typeAppParameters descriptor rather than sending it somewhere, it doesn't matter what target address we use. I just used the process serial number (PSN) of this application as a dummy value. Because this is an application and not something like a driver, I can just use the "kCurrentProcess" constant to represent this application's PSN. A descriptor is made of this PSN, then this PSN is used when creating the new kAEOpenDocuments AppleEvent. Once this Apple event is created, the PSN descriptor is no longer needed and is disposed of."

In other words, you just dummy up an address and coerce the descriptor to be of type typeAppParameters, and the address will be corrected for you by the AppleEvents Manager. Please refer to the ProcDoggie sample source code or to LaunchWithDoc (in the Snippets collection) for the complete picture of how this is done.

Apple event application error handling

Date Written: 7/22/91

Last reviewed: 10/15/99

What do we return to the Apple event handler if we get an application error while processing a standard event, Edition Manager event, or custom Apple event for commands and queries? Probably not errAENotHandled, since that means we didn't handle the event, which is different from trying to handle it and failing. Would it be errAEFail? What if we want to return more specific error information? Do we define our own errors, or try to use Apple's errors such as memFullErr or parmErr?

———

'errAEEventNotHandled' is not the correct value to return. In this case it tells the AppleEvent Manager that your routine did not handle the event and it should attempt to dispatch the event to a system event handler. As a result problems such as low memory may never be reported to the client. A better way to approach the problem is to never install handlers for events your application does not handle (so it never has to return 'errAEEventNotHandled'). And, always fill in 'errn' and 'errs' parameters in the reply event if anything went wrong (as shown on page 6-49 of *Inside Macintosh* Volume VI). You can be as descriptive as you like in the text--the more the better, in fact, since this text will be seen at the user level. The errn value you pass back should be the system error number that caused the exception. This value will then be passed back to the caller, and they can use this information for their own error recovery.

Date Written: 8/12/91

Last reviewed: 8/1/92

If I send Apple events to myself and specify kAEQueueReply to AESend, the event doesn't get put in the queue as I requested. It shows up immediately in the reply parameter. According to *Inside Macintosh,* if I specify kAEQueueReply I should treat the reply event as undefined. Please help; my application falls apart because it never receives the event it's supposed to. If this is a bug, will the behavior be changed in the future?

———

This isn't a bug; it's an undocumented "feature" of the Apple Event Manager (AEM). If you send an Apple event to yourself, the AEM directly dispatches to the handler you've installed for that event. So Apple events you send to yourself don't come in through WaitNextEvent. This means that if you reply to an Apple event you sent yourself, your 'ansr' handler will get

called directly.

This was not an arbitrary decision, though it can have some confusing ramifications for an application. Two factors influenced the decision--the first minor, the second major:

- Speed. The AEM has all the handlers for your application in a hashed table, and can dispatch very quickly to them, so for performance reasons direct dispatching was implemented.
- Event priorities and sequencing. Apple events have a lower priority than user-generated events (keystrokes, clicks); they come in right before update events. This created a potentially serious problem for applications that sent Apple events to themselves.

If all Apple events came through the event loop, you could easily create the following scenario:

1. The user selects a menu item, the application sends an Apple event to itself in response, and this Apple event requires a reply or will cause other Apple events to be sent.
2. The user clicks the mouse in an application window.

The mouse click has a higher priority than the reply or any Apple events that are sent in response to the first Apple event, and gets posted ahead of the Apple event in the event queue. This means that the mouse click happens and conceivably changes the current context of the application (perhaps switching windows, for example); then when the Apple events sent by the menu item handler are processed through the queue, the application state is not the same as it was when the menu selection was made, and the menu selection may be totally inappropriate for the current configuration.

So, to prevent a loss of sequencing, the AEM directly dispatches. Any non-Apple events that happen while you're sending and processing an Apple event to yourself will be queued and won't interrupt the Apple event process you've initiated. What this means in the case you're describing is that queued replies don't happen when you're sending to yourself. The AEM will directly dispatch to your 'ansr' handler, bypassing WaitNextEvent processing of Apple events to prevent any other events from breaking the chain of Apple events you may be processing. This isn't a major problem, but it's something you need to be aware of if you're expecting some other events to be processed before you get a reply or other Apple event.

Target Process must be valid and high-level event aware

Date Written: 8/16/91

Last reviewed: 6/14/93

When I send an 'odoc' Apple event to an already-running application, I get an -600 error at the AESend call. The process serial number is correct at entry. What is this error, and is it documented anywhere?

———

The error -600 means that the process serial number that you sent is either invalid or is the serial number of a process that does not accept Apple events--that is, the high-level event bit of the destination process's size resource isn't set.

LaunchApplication and event sequence

Date Written: 2/19/92

Last reviewed: 6/14/93

Is it true that if I double-click a document belonging to my application, the application will be launched and will receive an 'odoc' Apple event, but will not receive an 'oapp' event--that is, it will receive either 'odoc' or 'oapp' but not both?

———

Yes, except actually it will receive one of 'oapp', 'odoc', or 'pdoc'. The 'pdoc' will be followed (as the next event) by a 'quit' if the 'pdoc' was the event sent as the application was launched.

This is the normal sequence of events, and should be adhered to by everyone who launches applications. However, it isn't enforced by the system or the Finder. It's possible for any application to launch your application with any event, since it can stuff anything in the launchAppParameters field of LaunchApplication, as long as it's a valid high-level (not even

Apple) event. Launching another application this way would be bad programming, and would break most applications, but you should be aware that someone who doesn't understand event handling may do this to you.

Note that if another application launches your application using LaunchApplication and doesn't specify any high-level event in the launch parameter block, the Process Manager will automatically supply the 'oapp' event. So, in general, if Apple events and launching have been coded correctly, you'll always receive an 'oapp', 'odoc', or 'pdoc'.

Ignoring or purging an event from high-level event queue

Date Written: 3/6/92

Last reviewed: 6/14/93

Inside Macintosh Volume VI states that you cannot use FlushEvents to flush high-level events. I need to flush a single newFile event from the high-level event queue. How can I do this?

———

Typically, if you need to ignore a particular event which you would otherwise handle, you can just set a flag to tell the handler routine not to take its normal action. For most circumstances, this is the easiest and most appropriate way to ignore a particular event.

There may be circumstances where you need to purge a specific event from the high-level event queue. That can be done by searching for the event with GetSpecificHighLevelEvent and an appropriate filter proc; if the event is found, just call AcceptHighLevelEvent to dequeue it. Here's an example which removes all 'pdoc's from the high-level event queue:

```
TYPE
  EvtClassIDPtr = ^EvtClassID;
  EvtClassID = RECORD
    class: OSType;
    evtID: OSType;
  END;
VAR
  theEvtClassID: EvtClassID;
  retCode: OSErr;
  gotHLEFlag: Boolean;

  FUNCTION MyGSHLEFilter(myDataPtr: Ptr; msgHLEMPtr: HighLevelEventMsgPtr;
    sender: TargetID): BOOLEAN;
  VAR
    myTarg: TargetID;
    myRefCon: LongInt;
    myBuff: Ptr;
    myLen: LongInt;
    myErr: OSErr;
  BEGIN
    MyGSHLEFilter := FALSE;
    IF (OSType(msgHLEMPtr^.theMsgEvent.message) =
                                    EvtClassIDPtr(myDataPtr)^.class) AND
      (OSType(msgHLEMPtr^.theMsgEvent.where) =
                                    EvtClassIDPtr(myDataPtr)^.evtID) THEN
      BEGIN
        myLen := 0;
        myBuff := NIL;

        myErr := AcceptHighLevelEvent(myTarg, myRefCon, myBuff, myLen);
        IF myErr = bufferIsSmall THEN
```

```
        BEGIN
          myBuff := NewPtr(myLen);
          myErr := AcceptHighLevelEvent(myTarg, myRefCon, myBuff, myLen);
        END;
        IF myErr <> noErr THEN HandleError(myErr);
        IF myBuff <> NIL THEN DisposePtr(myBuff);


        MyGSHLEFilter := TRUE;
      END;
  END;
...
BEGIN     { main }
  ...
  { purge all print doc Apple events from the HLE queue }
  theEvtClassID.class := kCoreEventClass;
  theEvtClassID.evtID := kAEPrintDocuments;
  REPEAT
    gotHLEFlag := GetSpecificHighLevelEvent(@MyGSHLEFilter,
                                                @theEvtClassID, retCode)
  UNTIL (NOT gotHLEFlag) OR (retCode <> noErr);
  ...
END.
```

GetSpecificHighLevelEvent is documented in Chapter 5 of *Inside Macintosh* Volume VI.

Apple Event handler: Installation code

Date Written: 9/10/91

Last reviewed: 6/14/93

I'm writing a small application that uses Apple events, which neither opens documents nor prints. *Inside Macintosh* Volume VI says that all Apple event-aware applications should support the odoc and pdoc Apple events, but these are not appropriate in my case. What should I do?

You don't necessarily need to install a handler for each event, but it makes your code cleaner. If you don't have a handler, the Apple Event Manager will automatically return errAEEventNotHandled from your AEProcessAppleEvent call, since it can't find a handler for the event in the handler tables. But we recommend installing handlers for the required Apple events anyway; it makes your code cleaner and easier to understand, as well as easily allowing you a place to put a routine when you need to implement it. And it doesn't take up much memory. Just install a handler like this (in MPW C):

```
pascal OSErr AEPrintHandler(AppleEvent *messagein,
AppleEvent *reply, long refConIn)
{
/* Tell the compiler I'm not using these parameters */
#pragma unused (reply,refConIn,messagein)
    return(errAEEventNotHandled);  /* and return my error */
}
```

Finding Apple event sender's target ID or process serial number

Date Written: 10/25/91

Last reviewed: 6/14/93

How can I identify the sender of an Apple event?

If your application is just sending a reply, it should not be creating an Apple event or calling AESend. Instead, the Apple event handler should stuff the response information into the reply event, as shown on page 6-50 of *Inside Macintosh* Volume VI. The Apple Event Manager takes care of addressing and sending the event.

To find the target ID or process serial number of the sender of an Apple event, use AEGetAttributePtr to extract the address attribute, as follows:

```
retCode := AEGetAttributePtr(myAppleEvent, keyAddressAttr,
                typeWildCard, senderType, @senderBuffer,
                sizeof(senderBuffer), senderSize)
```

The senderBuffer can later be used with AECreateDesc to create an address to be passed to AESend. The buffer should be at least as large as data type TargetID. See *Inside Macintosh* Volume VI, page 5-22, for a description of TargetID.

Launching an application remotely

Date Written: 4/16/92

Last reviewed: 6/14/93

I need to launch an application remotely. How do I do this? The Process Manager doesn't seem to be able to launch an application on another machine and the Finder Suite doesn't have a Launch Apple event.

What you need to do is use the OpenSelection Finder event. Send an OpenSelection to the Finder that's running on the machine you want to launch the other application on, and the Finder will resolve the OpenSelection into a launch of the application.

As you can see if you glance at the OpenSelection event in the Apple Event Registry, there's one difficulty with using it for remote launching: You have to pass an alias to the application you want to launch. If the machine you want to launch the application on is already mounted as a file server, this isn't important, since you can create an alias to that application right at that moment. Or, if you've connected in the past (using that machine as a server) you can send a previously created alias and it will be resolved properly by the Finder on the remote machine.

However, if you want to launch a file without logging on to the other machine as a server, you'll need to use the NewAliasMinimalFromFullPath routine in the Alias Manager. With this, you'll pass the full pathname of the application on the machine you want to launch on, and the Alias Manager will make an alias to it in the same way it does for unmounted volumes. The obvious drawback here is that you'll need to know the full pathname of the application -- but there's a price to pay for everything. The FinderOpenSel sample code on the *Developer CD Series* disc illustrates this use of the NewAliasMinimalFromFullPath routine.

Opening documents from another application

Date Written: 5/2/91

Last reviewed: 6/14/93

How can I open Macintosh documents that belong to an application that's already running?

If you find that you have the Apple Event Manager present, then build 'odoc' Apple events to your heart's content, and send them to whomever you wish. The Process Manager will hop in there if the target isn't Apple event-aware and coerce the required Apple event to the appropriate Puppet String for you. No further hacks on your part are required. You literally don't care (at least for the purposes of the required Apple events) whether your target is Apple event aware or not.

Alert user instead of sending 'quit' Apple event to free RAM

Date Written: 5/2/91

Last reviewed: 6/14/93

Is it acceptable to force other Macintosh applications to quit to free up memory?

———

While it's possible to free up memory by sending a 'quit' Apple event, a much better alternative from a human interface standpoint is to pose an alert to the user indicating that you couldn't open the documents they requested due to a lack of memory, and asking the user to please try closing some windows or quitting some other applications and try again. It's better to leave the user in control of the process.

High-level Macintosh events and user reference numbers

Date Written: 4/24/92

Last reviewed: 7/13/92

We're using Apple events with the PPC Toolbox. We call StartSecureSession after PPCBrowser to authenticate the user's identity. The user identity dialog box is displayed and everything looks good. However, in the first AESend call we make, the user identity dialog is displayed again. (It isn't displayed after that.) Why is this dialog being displayed from AESend when I've already authenticated the user identity with StartSecureSession?

———

First, a few PPC facts:

- When a PPC session is started, StartSecureSession lets the user authenticate the session (if the session is with a program on another Macintosh) and returns a user reference number for that connection in the userRefNum field of the PPCStartPBRec. That user reference number can be used to start another connection (using PPCStart instead of StartSecureSession) with the same remote Macintosh, bypassing the authentication dialogs.

- User reference numbers are valid until either they're deleted with the DeleteUserIdentity function or one of the Macintosh systems is restarted.

- If the name and password combination used to start a session is the same as that of the owner of the Macintosh being used, the user reference number returned refers to the default user. The default user reference number normally is never deleted and is valid for connections to the other Macintosh until it's deleted with DeleteUserIdentity or one of the Macintosh systems is restarted.

With that out of the way, here's how user reference numbers are used when sending high-level events and Apple events: When you first send a high-level event or an Apple event to another Macintosh, the code that starts the session with the other system doesn't attempt to use the default user reference number or any other user reference number to start the session, and it doesn't keep the user reference number returned to it by StartSecureSession. The session is kept open for the life of the application, or until the other side of the session or a network failure breaks the connection.

When you started your PPC session, StartSecureSession created a user reference number that could be used to start another PPC session without authentication. However, the Event Manager knows nothing of that user reference number, so when you send your first Apple event, the Event Manager calls StartSecureSession again to authenticate the new session. Since there isn't any way for you to pass the user reference number from the PPC session to the Event Manager to start its session, there's nothing you can do about this behavior.

Sending applications must be high-level event aware

Date Written: 7/30/92

Last reviewed: 6/14/93

Why do I get error -903 (a PPC Toolbox noPortErr) when I send an Apple event to a running application with AESend?

———

The isHighLevelEventAware bit of the sending application's SIZE -1 resource (and SIZE 0 resource, if any) must be set.

Back to top

Downloadables

Acrobat version of this Note (K) Download

Back to top
