New Technical Notes

Macintosh

Developer Support

Apple Event Manager Q&As

Interapplication Communication **M.IC.AEMgr.Q&As**

Revised by: Developer Support Center October 1992
Written by: Developer Support Center October 1990

This Technical Note contains a collection of Q&As relating to a specific topic—questions you've sent the Developer Support Center (DSC) along with answers from the DSC engineers. While DSC engineers have checked the Q&A content for accuracy, the Q&A Technical Notes don't have the editing and organization of other Technical Notes. The Q&A function is to get new technical information and updates to you quickly, saving the polish for when the information migrates into reference manuals.

Q&As are now included with Technical Notes to make access to technical updates easier for you. If you have comments or suggestions about Q&A content or distribution, please let us know by sending an AppleLink to DEVFEEDBACK. Apple Partners may send technical questions about Q&A content to DEVSUPPORT for resolution.

New Q&As and Q&As revised this month are marked with a bar in the side margin.

Construction of Apple events
Written: 4/9/91
Last reviewed: 8/1/92

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

—

You can coerce the 'odoc' 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

Written: 7/22/91
Last reviewed: 8/1/92

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?

—

You pass back errAENotHandled, because it's true, and because some simple applications will not be able to handle anything more than that. What you can also do, and what most commercial applications will do (particularly applications that want to be scripting savvy), is add errn and errs parameters to the reply record for that event (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 usually. The errn value you pass back can be the system error number; then the sending program may be able to recover and try again.

Apple events to yourself bypass WaitNextEvent processing
Written: 8/12/91
Last reviewed: 8/1/92

If I send Apple events to myself and specify kAEQueueReply to AESend, the event does not get put in the queue like I asked for. 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 app falls apart because it never receives the event it is supposed to. If this is a bug, will the behavior be changed in the future?

—

This is not a bug, it's an undocumented "feature" of the Apple Event Manager. If you send an Apple event to yourself, the Apple Event Manager *directly dispatches* to the handler you have installed for that event. So Apple events you send to yourself do not come in through WaitNextEvent.

This means that if you reply to an Apple event you sent yourself, then your 'ansr' handler will get called directly. This was not an arbitrary decision, though it can have some confusing ramification for an application. Two factors influenced the decision, the first minor, the second major:
1) Speed. The Apple Event Manager 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.
2) Event priorities and sequencing. Apple events have a lower priority than user-generated

events (keys, clicks), Apple events 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, the following scenario could very easily be created:

• User selects menu item, 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.
• User clicks with the mouse in an application window.

The mouse click would have a higher priority than the reply or any Apple events that are sent in response to the first Apple event, and would get posted ahead of the Apple event in the event queue. Which would mean that the mouse click would happen and conceivably change the current context of the application (perhaps switching windows, for example); then when the Apple events sent by the menu item handler were processed through the queue, the application state would *not* be 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 Apple Event Manager (AEM) directly dispatches. Any non-Apple events that happen while you're sending and processing an Apple event to yourself will be queued and not interrupt the Apple event process you've initiated.

What this means in the case you're describing is that queued replies do not happen when you are sending to yourself. The AEM will directly dispatch to your 'ansr' handler, to prevent any other events from breaking the chain of Apple events you may be processing.

So WaitNextEvent processing of Apple events is bypassed when you are sending events to yourself. This isn't a major problem, but it is something you need to be aware of if you are expecting some other events to be processed before you get a reply or other Apple event.

X-Ref:
Macintosh Technical Note "Getting in Touch with Yourself via the Apple Event Manager"

Target Process must be valid and high-level event aware
Written: 8/16/91
Last reviewed: 8/1/92

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
Written: 2/19/92
Last reviewed: 8/1/92

Is it the case 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?

—

You’re correct, 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 Finder will automatically supply the 'oapp' event. So, in general, in the correct way to code Apple events and launching, you'll always receive an 'oapp', 'odoc', or 'pdoc'.

Ignoring or purging an event from high-level event queue
Written: 3/6/92
Last reviewed: 8/1/92

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

Written: 9/10/91
Last reviewed: 8/1/92

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

Written: 10/25/91
Last reviewed: 8/1/92

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, it should stuff the response information into the reply event in the Apple event handler, 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

Written: 4/16/92
Last reviewed: 5/21/92

I need to launch an application remotely. I have Apple event client code which needs to start up the server if it is not already executing. 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 does not have a Launch Apple Event.

—

What you need to do this is to use the OpenSelection Finder event. Send an OpenSelection to the Finder which is running on the machine you want to launch the other application, and the Finder will resolve the OpenSelection into a launch of the application. As you can see if you glance at the OpenSelction event in the registry, there is 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 wish 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 have connected in the past (using that machine as a server) you can use a previously created alias and you will just be prompted to log on to the server at launch time.

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 call in the Alias Manager. With this, you'll pass the full path name of the application on the other machine you wish to launch, 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 shows the NewAliasMinimalFromFullPath method.

Opening documents from another application

Written: 5/2/91
Last reviewed: 8/1/92

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
Written: 5/2/91
Last reviewed: 8/1/92

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 quitting some other applications and try again. It's better to leave the user in control of the process.

Sending applications must be high-level event aware
Written: 7/30/92
Last reviewed: 7/30/92

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

—

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

High-level Macintosh events and user reference numbers
Written: 4/24/92
Last reviewed: 7/13/92

We are using Apple events with the Program-to-Program Communications (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. The user identity dialog is not be displayed after that. Why is this dialog is being displayed from AESend when I've already authenticated the user identity with StartSecureSession?

First a few PPC facts:

1. 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.

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

3. If the name and password combination used to start a session is the same as the owner of the Macintosh the user is using, then the user reference number returned refers to the default user. The default user reference number is normally never deleted and is valid for connections to the other Macintosh until one of the actions in #2 above occurs.

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 does not attempt to use the default user reference number, or any other user reference number to start the session and it does not 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 which 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, it 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
Written: 7/30/92
Last reviewed: 9/15/92

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.