New Technical Notes
Macintosh

Developer Support

AppleTalk Overview Q&As
Networking **M.NW.ATOv.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.

How to get the User and Computer names used by AppleTalk
Written: 11/27/91
Last reviewed: 11/27/91

I'd like to use the same names that the system uses to identify itself on the AppleTalk network in my program. Where can I find those names?

___

The names used by the system for network services are stored in two 'STR ' resources in the System file. Your program can retrieve those names with the Resource Manager's GetString function.

Only one of the names is available in systems before System 7; the name set by the Chooser desk accessory. That name is stored in 'STR ' resource ID -16096. Under System 7, the Sharing Setup control panel lets the user assign two names for network services: the Owner name and the Computer name.

The Owner name is the name stored in 'STR ' resource ID -16096 and it identifies the user of the Macintosh. The Owner name is used by System 7 for two primary purposes: to identify the owner of the system when accessing the system remotely through System 7 File Sharing or the user identity dialog used by the PPC Toolbox (and Apple Event Manager), and as the default user name when logging on to other file servers with the Chooser.

The Computer name (also known as “Flagship name”) is the name stored in 'STR ' resource ID -16413 and used to identify the Macintosh. The Computer name is the name used by system network services to identify themselves on the AppleTalk network. For example, if your system’s Computer name were "PizzaBox", the PPC Toolbox registers the name “PizzaBox:PPCToolBox@*” when you start Program Linking and File Sharing registers the name “PizzaBox:AFPServer@*” when you start File Sharing.

Which AppleTalk routines can & can’t be called at interrupt time
Written: 8/8/91
Last reviewed: 10/9/91

Can any AppleTalk routines be called at interrupt time? Inside Macintosh says that DDPWrite and DDPRead can’t be called from interrupts. If all higher-level AppleTalk protocols are based on DDP, it seems that they all would not work.

___

The AppleTalk routines you can’t call at interrupt time are the original AppleTalk Pascal Interfaces listed in Inside Macintosh Volume II; these are also known as the “Alternate Interface” AppleTalk routines, or ABPasIntf.

The Alternate Interface routines cannot be called at interrupt time because they allocate the memory structures needed to make the equivalent assembly language AppleTalk call. For example, when the NBPLookup routine is called, it’s passed a handle to an ABusRecord. NBPLookup then has to allocate an MPPParamBlock and move the parameters from the ABusRecord into the newly allocated MPPParamBlock. Then NBPLookup makes a LookupName call, passing it the MPPParamBlock. When LookupName completes, NBPLookup must move results into the ABusRecord and release the memory used by the MPPParamBlock. Since memory is allocated and released within the routine, it cannot be called at interrupt time.

With that out of the way, the calls you can make at interrupt time (with some restrictions listed below) are what Apple calls the “Preferred Interface” AppleTalk routines. Most of the Preferred Interface routines are listed on page 562 of Inside Macintosh Volume V. There are a few additional calls that were added after the publication of Inside Macintosh Volume V; they’re documented in the AppleTalk chapter of Inside Macintosh Volume VI.

The Preferred Interface AppleTalk routines can be made at interrupt time as long as:

• You make them asynchronously with a completion routine (that is, the asynch parameter must be TRUE and you must provide a pointer to the completion routine in the ioCompletion field of the call’s parameter block). Making a call asynchronously and polling ioResult immediately afterward within the same interrupt-time code (which is basically the same as making the call synchronously) is not the same as using a completion routine.

• They are not listed as routines that may move or purge memory. The Preferred Interface routines do not allocate or dispose of any memory, since they’re just high-level ways to make the assembly language AppleTalk calls and are not built upon the old Alternate Interface routines.

X-Ref:
Macintosh Technical Note “Using the High-Level AppleTalk Routines”
Macintosh Technical Note “AppleTalk Interface Update”
Macintosh Technical Note “Avoid Use of Network Events”

AppleTalk and 'INIT's

Written: 9/5/91
Last reviewed: 9/17/91

Does AppleTalk work the same with an 'INIT' as with an application? I'm using only Name Binding Protocol (NBP) and Datagram Delivery Protocol (DDP) stuff.

___

Yes! Responder is an example of an 'INIT' that utilizes the NBP to register a Macintosh on the network. NBP requires DDP, so it is also present. AppleTalk is loaded before any 'INIT's, which makes it available to them.

Install AppleTalk with Installer instead of drag-installing

Written: 12/3/91
Last reviewed: 12/11/91

We have problems using AppleTalk version 56 on a Macintosh Plus under System 6. The problems go away when we drop AppleTalk version 52 into the System Folder. What's going on?

___

AppleTalk version 56 isn't working for you because you just copied the AppleTalk file into the System Folder of your Macintosh with the Finder (this is known as a "drag-install"). All versions of AppleTalk from version 53 up through the current version must be installed by an Installer Script. If you drag-install those versions of AppleTalk, several very important system resources won't be copied into the System file and AppleTalk won't work.

Apple software products that require AppleTalk version 53 or greater always include the Installer and Installer script that copy all system resources needed correctly. Apple also supplies the source to an Installer script that you can use if you license AppleTalk to ship with your products. See the "AppleTalk Licensing Disk 3.2" folder on the latest Developer CD for a look at the Installer script code. As usual, you'll need to contact Software Licensing at AppleLink address SW.LICENSE if you want to ship AppleTalk with your products.

Preferred AppleTalk calls and AppleTalk version

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

Do I need AppleTalk version 48 to call AppleTalk routines using the preferred AppleTalk interface?

___

The preferred AppleTalk routines don’t require AppleTalk version 48. You can make any of the AppleTalk calls listed in Inside Macintosh Volume II from a Macintosh Plus. To make .XPP driver calls listed in Inside Macintosh Volume V, you’ll need to use AppleTalk version 48. To make the new AppleTalk calls listed in Inside Macintosh Volume VI (and the Macintosh Technical Note "AppleTalk Phase 2 on the Macintosh"), you’ll need version 53 or greater. To make ADSP calls to the .DSP driver, you’ll need to either install ADSP or use AppleTalk version 56 or greater, which includes the .DSP driver.

AppleTalk Peek and Poke tools

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

Is there an update for the Apple tool called Poke?

___

The following versions of Peek and Poke are current:

AppleTalk Peek 3.1
AppleTalk Peek 3.2d5
AppleTalk Poke 3.1

You'll find them on the latest Developer CD Series disc, in Tools & Apps:Networking & Communications:AppleTalk Tools.

Using AppleTalk self-send mode

Written: 4/20/92
Last reviewed: 7/13/92

What's the recommended method of allowing an AppleTalk node to send packets to itself using AppleTalk's self-send mode (intranode delivery), assuming customers are running various versions of AppleTalk? There used to be a control panel called SetSelfSend that would turn AppleTalk self-send mode on at boot time. Should we use that control panel or should we use the PSetSelfSend function in our program to set the self-send flag ourselves?

___

AppleTalk self-send mode requires AppleTalk version 48 or greater. You can check the AppleTalk version with Gestalt or SysEnvirons. All Macintosh models except for the Macintosh XL, 128, 512, and Plus have AppleTalk version 48 or greater in ROM.

The SetSelfSend control panel is still available on the latest Developer CD (Tools & Apps (Moof!):Intriguing Inits/cdevs/DAs:Pete's hacks-Moof!:SetSelfSend). However, DTS doesn't recommend it as a solution if you need to use self-send mode in your program. Instead, you should use the PSetSelfSend function to turn self-send mode on with your program.

AppleTalk's self-send mode presents a problem. Any changes made to the state of self-send will affect another programs that use AppleTalk. That is, self-send mode is global to the system. Because self-send is a global setting, programs using self-send should follow these guidelines:

• If self-send is only needed for a brief period of time (for example, to perform a PLookupName on your own node), then you should turn self-send on with PSetSelfSend (saving the current setting returned in oldSelfFlag), make the call(s) that require self-send, and then restore self-send to its previous state.

• If self-send is needed for an extended period of time (for example, for the life of your application) where your program will give up time to other programs, then you should turn self-send on and leave it on—do not restore it to its previous state! Since other programs running on your system (that aren't well behaved) may turn off self-send at any time, programs that require self-send mode should periodically check to make sure self-send mode is still enabled with either the PSetSelfSend function or the PGetAppleTalkInfo function. Apple's system software has no compatibility problems with self-send—that is, it doesn't care if it's on or off—so leaving it on won't hurt anything.