New Technical Notes
Macintosh

Developer Support

AppleTalk Interface Update
Networking **M.NW.AppleTalkInterfaceUpdate**

Revised by: March 1988
Written by: Bryan Stearns July 1987

M.NW.HL.AppleTalk announced that we would be moving to a simplified AppleTalk Manager interface. That interface is available now, as part of MPW 2.0 and newer.

Documentation for this new interface is contained in the AppleTalk Manager chapter of *Inside Macintosh Volume V*. This technical note contains some of the preliminary documentation for this interface and some useful points about information about it, and AppleTalk in general.

The original AppleTalk Pascal Interfaces, known as `ABPasIntf`, were designed to simplify use of AppleTalk from high-level languages. Instead, they've caused us a few compatibility problems. We've decided to encourage use of the same interface that assembly-language AppleTalk uses, a parameter-block interface in the same style as the low-level interfaces to the File and Device Managers.

The original calls are still supported (and will be for a while) as an "alternate" interface, but we suggest that you consider moving to the new "preferred" calls. Be warned that use of the original calls may cause compatibility problems with future system software. Also, new protocols (like ASP, the AppleTalk Session Protocol) are only provided with the new interfaces.

The new interface uses parameter blocks like those used by the File and Device Managers; you fill out the call-specific fields of the block, and a small amount of glue code (provided with development environments like MPW) turns the parameter block into a `Control` call to the appropriate AppleTalk driver.

Most calls have an interface like:

```
FUNCTION PSomeCall(thePBPtr: ATPPBptr; asyncFlag: BOOLEAN): OSErr;
```

The glue fills in the fields `csCode` and `ioRefNum` with the appropriate value for the call you're making.

Synchronous and Asynchronous calls

You can still make calls synchronously (“do it now”) or asynchronously (“start it now, finish it soon”). If you choose to make a call asynchronously, be sure to provide a completion routine in the `ioCompletion` field (to be called when the call finally finishes), or poll the `ioResult` field of the parameter block (the call is done if `ioResult` is less than or equal to 0).

You must not move or dispose of a parameter block before the call finishes; when the call does complete, you are responsible for throwing the parameter block away (if you allocated it using Memory Manager routines).

Note that the alternate interfaces generated a network event on completion of an asynchronous call; this service is not provided by the preferred interfaces, partly because of future compatibility problems. See M.NW.NoNetEvents for background information.

Packed data structures

Several of the data structures used by the new interfaces are packed; Pascal doesn't deal well with these structures. Special calls are provided for building LAP and DPP write-data structures, NBP names-table elements, and ATP buffer data structures.

For example, when registering a name (using `PRegisterName)`, you'll use a `NamesTableEntry` structure. This structure consists of a few unpacked fields, followed by an entity-name: three strings (representing the object, type, and zone fields of the name) packed together. You can call `NBPSetNTE` to pack the strings into the `NamesTableEntry` structure. When you remove the name (`PRemoveName`), you'll use the entity-name by itself; you can use `NBPSetEntity` to pack it in.

Zone Interface Protocol

A function, `GetBridgeAddress`, is provided to obtain the node ID of a bridge, for use in ZIP transactions (zero is returned if no bridge is present on your network). You make ZIP calls using ATP requests, as described in the *Inside AppleTalk* chapter on ZIP.

Further Reference:

- The AppleTalk Manager
- *Inside AppleTalk* (for ZIP information)
- M.NW.HL.AppleTalk
- M.NW.NoNetEvents