The .XPP Driver

Implementing multiple protocols

The Extended Protocol Package (XPP) is intended to implement several AppleTalk communication protocols in the same package for ease of use. The .XPP driver operates on two levels: the low-level module implements the workstation side of AppleTalk Session Protocol, and the high-level module implements a small portion of the workstation side of the AppleTalk Filing Protocol.

This driver adds functionality to the **AppleTalk Manager** by providing services additional to those provided in the .MPP and .ATP drivers.The .XPP driver maps an AFP call from the client workstation into one or more ASP calls. .XPP provides one client-level call for AFP.

The implementation of AFP in the .XPP driver is very limited. Most calls are a very simple one-to-one mapping from an AFP call to an ASP command without any interpretation of the syntax of the AFP command by the .XPP driver.

The .XPP driver supports ASP Version 0x0100, as described in *Inside AppleTalk*.

Error Reporting

Errors are returned by the .XPP driver in the ioResult field of the parameter block passed with **Control** calls to the **Device Manager** .

The error conditions reported by the .XPP driver may represent the unsuccessful completion of a routine in more than just one process involved in the interaction of the session. System-level, .XPP driver, AppleTalk, and server errors can all turn up in the ioResult field.

AFP calls return codes indicating the unsuccessful completion of AFP commands in the Command Result field of the parameter block.

An application using the .XPP driver should respond appropriately to error conditions reported from the different parts of the interactions. The following errors can be returned in the ioResult field:

1. System-level errors

System errors returned by the .XPP driver indicate such conditions as the driver not being open or a specific system call not being supported.

2. XPP errors

The .XPP driver can also return errors resulting from its own activity (for example, the referenced session is not open). The possible .XPP driver errors returned are listed in the .XPP driver results codes section with each function that can return the code.

3. AppleTalk Errors (returned from lower-level protocols)

.XPP may also return errors from low-level protocols (for example, "Socket not open").

4. An ASP-specific error could be returned from an ASP server in response

to a failed **OpenSession** call. Errors of this type, returned by the server to the workstation, are documented here under the .XPP results code section and in *Inside AppleTalk*, section 11, "AppleTalk Session Protocol".

5. The AppleTalk Filing Protocol defines errors that are required from the server to the workstation client. These errors are returned in the cmdResult field of the parameter block. At the ASP level, the cmdResult field is client defined data and may not be an error code.

Using AppleTalk Name Binding Protocol

A server wishing to advertise its service on the AppleTalk network calls ATP to open an ATP responding socket known as the session listening socket (SLS). The server then calls the Name Binding Protocol (NBP) to register a name on this socket. At his point, the server calls the server side of ASP to pass it the address of the SLS. Then, the server starts listening on the SLS for session opening requests coming over the network

Opening and Closing Sessions

When a workstation wishes to access a server, the workstation must call NBP to discover the SLS for that server. Then the workstation calls ASP to open a session with that server.

After determining the SLS (address) of the server, the workstation client issues an **OpenSession** (or **AFPLogin**) call to open a session with that server. As a result of this call, ASP sends a special **OpenSession** packet (an ATP request) to the SLS; this packet carries the address of a workstation socket for use in the session. This socket is referred to as the workstation session socket (WSS). If the server is unable to set up the session, it returns an error. If the request is successful, the server returns no error, and the session is opened. The open session packet also contains a version number so that both ends can verify that they are speaking the same version of ASP.

The **AbortOS** function can be used to abort an outstanding **OpenSession** request before it has completed.

The workstation client closes the session by issuing a **CloseSession** (or **AFPLogout**). The **CloseSession** call aborts any calls that are active on the session and closes the session. The session can also be closed by the server or by ASP itself, such as when one end of the session fails. The **CloseAll** call (which should be used with care) aborts every session that the driver has active.

Session Maintenance

A session will remain open until it is explicitly terminated by the ASP client at either end or until one of the sessions ends, fails, or unreachable.

Commands on an Open Session

Once a session has been opened, the workstation client can send a sequence of commands over the session to the server end. The commands are delivered in the same order as they are issued at the workstation end, and replies to the commands are returned to the workstation end.

Three types of commands can be made on an open session. These commands are

UserCommand, UserWrite, and AFPCall functions.

UserCommand calls are similar to ATP requests. The workstation client sends a command (included in a variable size command block) to the server client requesting it to perform a particular function and send back a variable size command reply. Examples of such commands vary from a request to open a particular file on a file server, to reading a certain range of bytes from a device. In the first case, a small amount of reply data is returned; in the second case a multiple-packet reply might be generated.

The .XPP driver does not interpret the command block or in any way participate in executing the command's function. It simply conveys the command block, included in a higher-level format, to the server end of the session, and returns the command reply to the workstation-end client. The command reply consists of a four-byte command result and a variable size command reply block.

UserWrite allows the workstation to convey blocks of data to the server. UserWrite is used to transfer a variable size block of data to the server end of the session and to receive a reply.

The AFPCall function provides a mechanism for passing an AFP command to the server end of an open session and receiving a reply. The first byte of the AFPCall command buffer contains the code for the AFP command that is to be passed to the server for execution. Most AFP calls are implemented through a very simple one-to-one mapping that takes the call and makes an ASP command out of it.

The AFPCall function can have one of four different, but very similar, formats.

Getting Server Status Information

ASP provides a service to allow its workstation clients to obtain a block of service status information from a server without the need for opening a session. The GetStatus function returns a status block from the server identified by the indicated address. ASP does not impose any structure on the status block. This structure is defined by the protocol above ASP.

Attention Mechanism

Attentions are defined in ASP as a way for the server to alert the workstation of some event or critical piece of information. The ASP **OpenSession** and **AFPLogin** calls include a pointer to an attention routine in their parameter blocks. This attention routine is called by the .XPP driver when it receives an attention from the server and also when the session is closing as described below.

In addition, upon receiving an **OpenSession** call or **AFPLogin** call, the .XPP driver sets the first two bytes of the session control block (SCB) to zero. When the .XPP driver receives an attention, the first two bytes of the SCB are set to the attention bytes from the packet (which are always nonzero).

Note: A higher-level language such as Pascal may not wish to have a low-level attention routine called. A high-level language program can poll the attention bytes, and if they are ever nonzero, the program will know that an attention has come in. (It would then set the attention bytes back to zero.) Of course, two or more attentions could be received between successive polls, and only the last one would be recorded.

The .XPP driver also calls the attention routine when the session is closed by either the server, workstation, or ASP itself (if the ASP session times out). In these cases, the attention bytes in the SCB are unchanged.

The Attention Routine

The attention routine is called at interrupt level and must observe interrupt conventions. Specifically, the interrupt routine can change registers A0 through A3 and D0 through D3 and it must not make any **Memory Manager** calls.

It will be called with:

- register D0 (word) equal to the SessRefnum for that session (see **OpenSession)**
- register D1 (word) equal to the attention bytes passed by the server (or zero if the session is closing)

Return with an RTS (return from subroutine) to resume normal execution.