HP-JetSend Session Protocol ERS

Version 3.00

Allen Baker

Hewlett-Packard Company
INSO R&D Lab

September 16, 1996

Table of Contents

1. Revisions 4
1.1 Version 3.00 (16sep96) 4
1.2 Version 2.02 (6sep96) 4
1.3 Version 2.01 (19aug96) 4
1.4 Version 2.00 (5jun96) 4
1.5 Version 1.10 (3jun96) 4
1.6 Version 1.00 4
2. Patents 5
3. Introduction 6
4. References 7
5. Interface 8
6. Protocol 9
6.1 Protocol Operation 9
6.1.1 OSP Session Management 9
6.1.1.1 Session States 10
6.1.1.1.1 CLOSED 10
6.1.1.1.2 LISTEN 10
6.1.1.1.3 CONNECTING 10
6.1.1.1.4 SYN-WAIT 11
6.1.1.1.5 SYN-SENT 11
6.1.1.1.6 OPEN 11
6.1.1.2 Opening a Session 11
6.1.1.2.1 Passive Session Opens 11
6.1.1.2.2 Active Session Opens 12
6.1.1.3 Managing an Open Session 13
6.1.1.4 Closing a Session 14
6.1.1.5 Detecting an Half-Open Session 14
6.1.2 Channel Management 15
6.1.2.1 Opening a Channel 15
6.1.2.2 Closing a Channel 15
6.1.2.3 Data Communication 16
6.1.2.4 OSP Fragmentation and Reassembly 16
6.1.3 Extracting OSP Messages From Stream Connections 16
6.1.3.1 Reassembling Messages 17
6.1.3.2 Seperating Messages 18
6.1.4 Gateways 18
6.1.4.1 Gateway Broadcasts 19
6.1.4.1.1 Gateway Broadcast Time-out 20
6.2 OSP Messages and Formats 21
6.2.1 OSP Header Format 21
6.2.2 OSP Header Fields 21
6.2.2.1 Control Bits 21
6.2.2.2 Version Number 22
6.2.3 Extended Header Format 22
6.2.4 Extended Header Fields 22
6.2.4.1 Extended Control Bits 22
6.2.5 SYN Message (0x01) 22
6.2.5.1 Message Format 23
6.2.5.2 Message Fields 23
6.2.5.2.1 Source Session 23
6.2.5.2.2 Passive OSP Port Number 23
6.2.5.2.3 OMN Form Target Address Length 23
6.2.5.2.4 OMN Form Target Address 24
6.2.6 ASYN Message (0x03) 24
6.2.6.1 Message Format 24
6.2.6.2 Message Fields 24
6.2.6.2.1 Source Session 24
6.2.6.2.2 Destination Session 24

6.2.7 NSYN Message (0x05) 25
6.2.7.1 Message Format 25
6.2.7.2 Message Fields 25
6.2.7.2.1 Destination Session 25
6.2.8 RST Message (0x08) 25
6.2.8.1 Message Format 26
6.2.8.2 Message Fields 26
6.2.8.2.1 Destination Session 26
6.2.9 NUL Message (0x10) 26
6.2.9.1 Message Format 26
6.2.9.2 Message Fields 27
6.2.9.2.1 Destination Session 27
6.2.10 CHN Message (0x20) 27
6.2.10.1 Message Format 27
6.2.10.2 Message Fields 27
6.2.10.2.1 Destination Session 27
6.2.10.2.2 Destination Channel 28
6.2.10.2.3 Data Length 28
6.2.10.2.4 Data 28
6.2.11 CSYN Message (0x21) 28
6.2.11.1 Message Format 28
6.2.11.2 Message Fields 28
6.2.11.2.1 Destination Session 28
6.2.11.2.2 Source Channel 28
6.2.12 CASYN Message (0x23) 29
6.2.12.1 Message Format 29
6.2.12.2 Message Fields 29
6.2.12.2.1 Destination Session 29
6.2.12.2.2 Source Channel 29
6.2.12.2.3 Destination Channel 29
6.2.13 CNSYN Message (0x25) 30
6.2.13.1 Message Format 30
6.2.13.2 Message Fields 30
6.2.13.2.1 Destination Session 30
6.2.13.2.2 Destination Channel 30
6.2.14 CRST Message (0x28) 30
6.2.14.1 Message Format 31
6.2.14.2 Message Fields 31
6.2.14.2.1 Destination Session 31
6.2.14.2.2 Destination Channel 31
6.2.15 CEXTLSN Message (0xA0) 31
6.2.15.1 Message Format 32
6.2.15.2 Message Fields 32
6.2.15.2.1 Destination Session 32
6.2.15.2.2 Message data bits. 32
6.2.16 GWY Message (0x40) 32
6.2.16.1 Message Format 33
6.2.16.2 Message Fields 33
6.2.16.2.1 OMN Form Target Address Length 33
6.2.16.2.2 OMN Form Target Address 33
6.2.17 GACK Message (0x42) 33
6.2.17.1 Message Format 34
6.2.17.2 Message Fields 34
6.2.17.2.1 OMN Form Gateway Address Length 34
6.2.17.2.2 OMN Form Target Address Length 34
6.2.17.2.3 OMN Form Addresses 34

1. Revisions

1.1 Version 3.00 (16sep96)

- Added EXT bit to header control bits
- Added Extended Header which includes an extra byte of control bits.
- Added CEXTLSN message for OSP to tell its peer that it is listening for channels.

1.2 Version 2.02 (6sep96)

- Changed ASYN timeout from (4*RoundTripTime) to max(1minute,(4*RoundTripTime)).
- Changed message format pictures to better show bit and byte positions.

1.3 Version 2.01 (19aug96)

- Added section on extracting OSP messages from stream connections.
- Expanded section on Gateways.
- Replaced use of "oipHandle" with "OSP port number".
- Specified that OMN form addresses are stored as zero terminated ASCII strings in OSP messages.
- Specified that OMN form address lengths include the terminating zero byte in the address.

1.4 Version 2.00 (5jun96)

- Added ability to locate gateways.
- Added ability for OSP to have multiple simultaneous sessions between one pair of devices (needed for gateways).
- Added session fields to messages.
- Removed obsolete references to OSCP.
- Improved wording in places.

1.5 Version 1.10 (3jun96)

- Changed value of Version field from 1 to 0.
- Changed session establishment time out time.
- Added description of keep-alive messages in Detecting a Half-Open Session section.
- Added section on Fragmentation and Reassembly.
- Removed Data Length, Source Channel Handle and Destination Channel Handle fields from header.
- Changed size of oipHandle and Channel fields from 32 bit to 16 bit.
- Changes size of Version field from 8 to 4 bits.
- Added 4 bits to Control Bits field.
- Defined new Control Bit - Last Fragment (LF)
- Rewrote channel management to simplify it.
- Improved descriptions of session open and close.

1.6 Version 1.00

- Initial version.

2. Patents

The protocol described in this document is protected under the following international patents which are owned by the Hewlett-Packard Company:

1. DE69802792D1
2. DE69802792T2
3. DE69835314D1
4. DE69835314T2
5. EP0872991A2
6. EP0872991A3
7. EP0872991B1
8. EP0886411A2
9. EP0886411A3
10. EP0976058A1
11. EP0976058B1
12. US6202096 B1
13. US6502000
14. US6721286
15. WO1998047076A1

3. Introduction

An External Reference Specification (ERS) specifies its target object's look, feel, and functionality from an external perspective. The external perspective is a view of the target object from the outside and does not include requirements specifications, the target object's internal architectural and design details, or the target object's architectural relationships with other subsystems and modules (the target object's relationships to other subsystems and modules is described in the Architecture Specification). The external perspective includes all of the target object's hardware and software interfaces, protocols, performance characteristics, functionality, and operational procedures. Interfaces are inputs and outputs like those to and from users and other systems, subsystems and modules. An ERS describes its target object's interfaces and functionalities as a whole, without describing how those interfaces and functionalities are allocated to subsystems and modules within the target object.

The target object for this ERS is the HP-JetSend Session Protocol layer (OSP). This ERS specifies OSP’s interface and its protocol.

4. References

[1] Kristin Dahl. HP-JetSend Session Protocol API. Hewlett-Packard Company, INSO R&D. Boise, ID. 1996.

[2] Patrick Arnold. HP-JetSend Gateway Architecture. Hewlett-Packard Company, INSO R&D. Boise, ID. 1996.

[3] Allen Baker. HP-JetSend Network ARS. Hewlett-Packard Company, INSO R&D. Boise, ID. 1996.

5. Interface

The service interface to OSP is its API. The API provides the HP-JetSend Interaction Protocol layer (OIP) with OSP services.

The OSP API is specified in a separate document - [1].

6. Protocol

6.1 Protocol Operation

6.1.1 OSP Session Management

OSP is a session-oriented protocol within which two processes may open multiple full-duplex communication channels between themselves. Messages from a sender are directed to an OSP port on the destination host.

OSP session management is implemented using the Reliable Message Transport Protocol (RMTP) which is a required part of all HP-JetSend implementations.

OSP STREAM channels may be implemented using any reliable, ordered delivery transport protocol such as TCP, TTP, or SPX. OSP MESSAGE channels are implemented using RMTP.

6.1.1.1 Session States

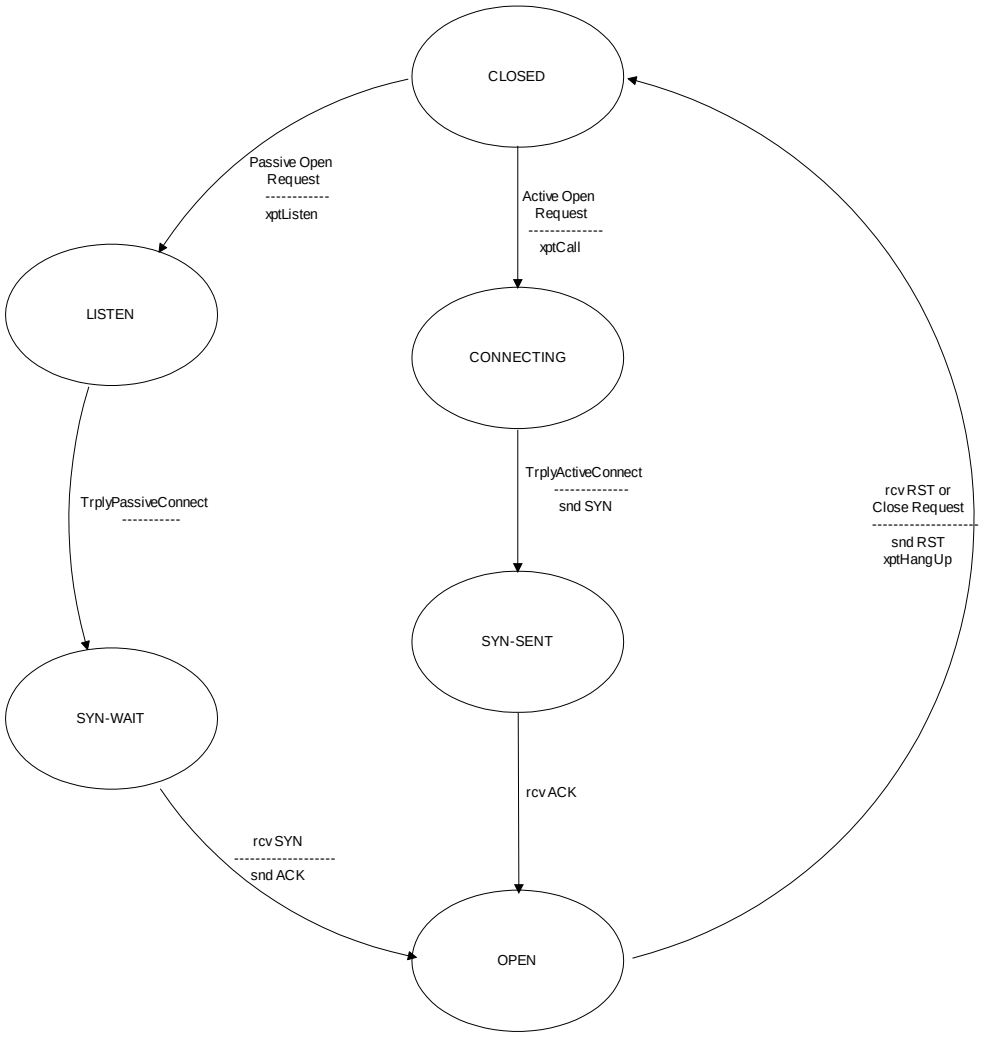


Figure 1

Each OSP session will progress through a series of states during its lifetime. The states are shown in **Figure 1** and are individually described below. In **Figure 1,** the ellipses represent the states of the OSP finite state machine and the arcs represent changes in state. Each arc is annotated with the event causing the state change and the resulting output.

6.1.1.1.1 CLOSED

The CLOSED state exists when no session exists.

6.1.1.1.2 LISTEN

The LISTEN state is entered after a passive Open request is processed. OSP waits for an active request to establish a transport connection from a remote host.

6.1.1.1.3 CONNECTING

The CONNECTING state is entered after a active Open request is processed. OSP attempts to actively open a transport connection to a remote host.

6.1.1.1.4 SYN-WAIT

The SYN-WAIT state is entered after passively opening a transport connection. OSP then waits in the SYN-WAIT state for a SYN message to arrive from the remote host.

6.1.1.1.5 SYN-SENT

The SYN-SENT state is entered after actively opening a transport connection. A SYN message is sent to the remote host. OSP then waits in the SYN-SENT state for acknowledgment (ASYN message) of its Open request.

6.1.1.1.6 OPEN

The OPEN state may be reached from either the SYN-WAIT or the SYN-SENT state. It is reached from the SYN_WAIT state when a SYN message is received from the remote host. It is reached from the SYN_SENT state when an ASYN for the SYN is received from the remote host.

In the Open state channels may be opened and data may be sent between the two parties of the session.

6.1.1.2 Opening a Session

Sessions are opened by issuing the Open request, which can be either active or passive. A passive Open request puts OSP into the LISTEN state, during which it passively listens for a request to open a session from a remote host. The active Open request attempts to establish a session with a specified OSP port at a remote host.

OSP requires the side that actively opened the transport connection to also actively open the session. The session open negotiation exists to exchange session identifiers and prevent sessions from being created when a non-HP-JetSend network entity accidentally opens a transport connection with HP-JetSend.

6.1.1.2.1 Passive Session Opens

OIP, identifying itself with its assigned OSP port number (variously referred to as HP-JetSend device number, HP-JetSend sub-address, and oipHandle), makes a request to OSP to listen for a session (a passive Open request). OSP responds by making a call to xptListen() using the MESSAGE service type (for RMTP connections). OSP, remembering the OSP port number that goes with the xptListen() call, waits for the xptListen() to generate TrplyPassiveConnect replies. Each TrplyPassiveConnect contains a unique connection handle (connHandle). For each resulting TrplyPassiveConnect, OSP creates a new session control block (SCB) and stores the OSP port number and the connHandle in it.

With the connection established and the SCB set-up, OSP then waits for the active side to send it a SYN message. If the first arriving message is not a SYN message or a SYN message does not arrive within 2 minutes, OSP closes the connection and destroys the SCB.

If the first message is a SYN message and it arrives within 2 minutes, OSP sends an ASYN or a NSYN over the connection to the remote host. OSP sends an ASYN to accept the session if the SYN message specified the same OSP port number as the one stored in the SCB. It sends a NSYN to reject the session otherwise.

If OSP is supporting multiple OIPs, and more than one OIP requests that OSP listen for a session on the same network, OSP only calls xptListen() on that network once. It maintains a list of the OSP port numbers for which it is listening on that network. When a transport connection is created from the xptListen() and the OSP SYN is received, OSP will reject the session request if the OSP port number in the SYN does not match any of those in the list.

If OSP accepted the session (ASYN), it leaves the connection open and returns a session handle to OIP.

If OSP rejected the session (NSYN), it waits until the NSYN is successfully sent to the remote host and then closes the connection and destroys the SCB.

6.1.1.2.2 Active Session Opens

OIP, identifying itself with an OSP port number and its target with an address string (which the OSP implementation converts to an OMN form address), makes a request to OSP to open a session (an active Open request). OSP responds by making a call to xptCall() using the MESSAGE service type (for RMTP connections). OSP, remembering the OSP port number that goes with the xptCall() call, waits for the xptCall() to generate a TrplyActiveConnect reply. Each TrplyAciveConnect contains a unique connection handle (connHandle). For the resulting TrplyActiveConnect, OSP creates a new session control block (SCB) and stores the OSP port number and the connHandle in it.

With the connection established and the SCB set-up, OSP then sends a SYN message to the passive side. The SYN message contains the passive OSP port number, which OSP derived from the target's OMN form address. OSP measures the amount of time that it takes for RMTP to confirm that it successfully sent the SYN message (RoundTripTime). It then waits for the passive side to send it a ASYN or NSYN message. If the first arriving message is not an ASYN or NSYN message, OSP closes the connection and destroys the SCB. If the ASYN or NSYN does not arrive within the larger of 1 minute or 4 times the RoundTripTime, OSP closes the connection and destroys the SCB. In psuedo-code this algorithm looks like:

```
Send(SYN)
Wait(max(1minute,(4*RoundTripTime)))
if ((first msg not ASYN ) or (ASYN has not arrived))
   Close(SESSION)
   Destroy(SCB)
```

Waiting for 4*RoundTripTime ensures that the responding side has plenty of time to respond over a slow link. Waiting 1 minute ensures that the responding side has plenty of time to respond if it is a relatively slow machine and is communicating over a fast link.

If the remote host accepted the session (ASYN), OSP leaves the connection open and returns a session handle to OIP.

If the remote host rejected the session (NSYN), OSP closes the connection and destroys the SCB.

6.1.1.3 Managing an Open Session

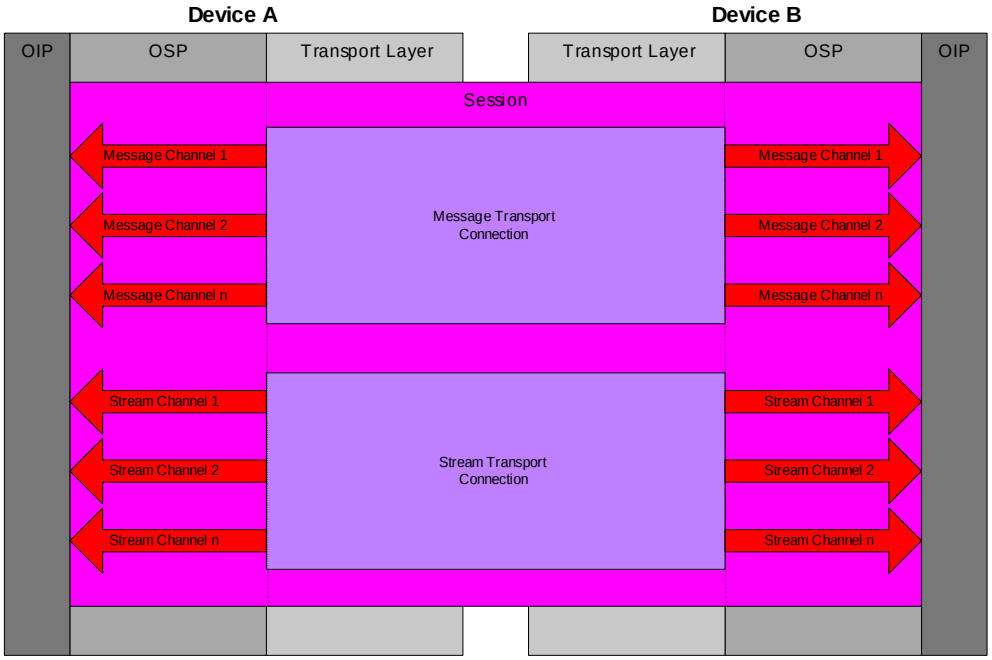


Figure 2

For any one session, OSP will maintain no more than one message (RMTP) connection with the remote host and no more than one stream (TCP, SPX, etc.) transport connection with the remote host. As **Figure 2** illustrates, OSP will multiplex all MESSAGE Channels over the single message connection and all STREAM Channels over the single stream connection.

After the session is established, an RMTP connection already exists. It gets created during the process of establishing the session. If OIP makes a request to OSP to open a MESSAGE channel, OSP negotiates the channel into existence over the RMTP connection.

When OIP makes its first request to passively listen for STREAM channels, OSP must first post a listen on its local well known stream transport port. When the remote host makes a transport connection to that port, OSP can then listen for a request to negotiate a channel into existence. Thereafter, when OIP requests to passively or actively open a STREAM channel, OSP will negotiate the channel into existence over that same transport connection.

When OIP makes its first request to actively open a STREAM channel, OSP must first open a stream transport connection to the well known stream transport port on the remote host. If the remote host is listening for a connection on that port, the connection is established and OSP can negotiate a channel into existence over it. Thereafter, when OIP requests to passively or actively open a STREAM channel, OSP will negotiate the channel into existence over that same transport connection.

If OIP first makes a request to passively listen for a STREAM channel, then later OIP makes a request to actively open a STREAM channel within the same session - while OSP is still passively listening for a transport connection - OSP will cancel its outstanding transport listen request before placing its transport

open request. This will prevent the possibility of two stream transport connections within the same session.

When all the channels on the stream transport connection are closed, OSP will close the connection. If OIP had earlier requested that OSP listen for a STREAM channel, OSP will reissue the listen request for a stream transport connection.

The RMTP connection is never closed during the lifetime of the session.

6.1.1.4 Closing a Session

When OSP closes a session, it must close all of the transport connections it has open within the session and cancel any transport listens that are active. OSP will close a session in response to 3 events:

1) It receives a Close request from OIP,
2) It receives a RST message from the remote host,
3) Its RMTP connection fails.

If it has received a Close session request from OIP, it sends a RST message over its RMTP connection, closes all its transport connections, cancels all its listens, and destroys the SCB. This simple session closing facility requires that users determine that all data has been reliably delivered before requesting a close of the session.

If it has received a RST message over its RMTP connection, it closes all its transport connections, cancels all its listens, and destroys the SCB.

If its RMTP connection fails, it closes all its transport connections, cancels all its listens, and destroys the SCB.

6.1.1.5 Detecting an Half-Open Session

If one side of a session crashes, the session may be left with the other side still active. This situation is termed to be an half-open session.

If a side of the OSP session is regularly sending messages to the other side, then the sending side may detect the half-open session during the normal course of sending messages. When it tries to send a message through its MESSAGE (RMTP) connection to the side that crashed, the transport connection will fail and OSP will close the session.

If it's the stream transport connection that fails, OSP will not automatically close the session. When this happens, OSP will send a keep-alive message over its RMTP connection to either verify that the remote host is still alive or force the RMTP connection to fail and close the session.

If a side is not regularly sending messages to the other side (as in the case of a printer that usually just passively waits for messages), that side will not detect a half-open session by a transport connection failing.

To ensure that unusable sessions/connections do not remain open and consuming resources, OSP will send keep-alive messages during long periods of inactivity in the session. If there has been no activity

on any channel within the session for longer than 10 minutes, OSP will send a keep-alive message (NUL message) to the other side over the RMTP connection. If the RMTP connection fails (OSP receives a TrplyDisconnect), the session is terminated. To prevent keep-alive messages from piling up in the RMTP layer, OSP will not send another keep-alive message until it receives a TrplySend to the last one it sent. (See Persist Timer section of RMTP ERS to understand conditions under which a good connection can take a long time to send a message).

6.1.2 Channel Management

OSP provides the abstraction of channels running through a session. When the session protocol is in the OPEN state, OSP may create STREAM and MESSAGE channels.

Each OSP channel acts as a full-duplex communication channel between two HP-JETSEND devices. Messages from a sender are directed to a channel end-point on the destination host.

6.1.2.1 Opening a Channel

Channels are opened by issuing the Open request, which can be either active or passive. A passive Open request causes OSP to passively listen for a request to open a channel from a remote host. The active Open request sends a request to open a channel to a remote host.

A channel open request is carried in a SYN message with the CHN bit also set (CSYN message). It includes the channel number from the sending side. The response to the CSYN is a SYN message with the CHN and ACK bits also set (CASYN message). It contains the channel number from the responding side.

The passive side of a channel open receives a CSYN message from the active side. The passive side records the active side's channel number in a channel control block (CCB). It then sends a CASYN back to the active side. It includes its own channel number (a handle to its CCB) in the CASYN. From that point forward, all messages that the passive side sends to the active side will contain the active side's channel number in the Destination Channel field.

The active side of a channel open creates a CCB and sends a CSYN message to the passive side. It includes its own channel number (a handle to its CCB) in the CSYN. It then receives a CASYN back from the passive side. The active side records the passive side's channel number in its CCB. From that point forward, all messages that the active side sends to the passive side will contain the passive side's channel number in the Destination Channel field.

The passive side may respond to a CSYN with a SYN message with the CHN and NAK bits also set (CNSYN message). This is a rejection of the channel open. No channel is created.

6.1.2.2 Closing a Channel

Closing a channel can be initiated by a Close request from the user process or by receipt of an CRST message from the other end of the channel. In the case of the Close request, OSP will send a CRST message to the other side of the channel and then closes its end of the channel. OSP discards messages directed to a closed channel. This simple channel closing facility requires that users determine

that all data has been delivered before requesting a close of the channel.

6.1.2.3 Data Communication

Data flows through an OSP channel in the form of CHN messages. All CHN messages have the LF bit set except for non-terminal fragments. See section 6.1.2.4 for usage of LF bit.

At the OSP level, outgoing messages, as they are created, are queued as input to the transport layer. Each message is reliably delivered by the underlying transport. If it cannot be delivered, the transport will generate a disconnect event on that connection, which will cause all channels running over that connection to close.

6.1.2.4 OSP Fragmentation and Reassembly

When OSP opens a connection, one of the items of information that it receives back from TIAPI in the TrplyActiveConnect (or TrplyPassiveConnect) is the maximum message size for that connection. The maximum message size is the size of the largest message that the transport will accept for sending from OSP on that connection. If OSP receives from OIP a request to send a buffer of data that - with an OSP header prepended - is larger than the maximum message size, OSP will break the data up into multiple OSP messages. It prepends its header to each of the multiple messages, sets the Last Fragment bit (LF) to 0 on all but the last one which it sets to 1, and sends them to TIAPI for transmission.

The receiving side reassembles the messages into one block of data for delivery to OIP.

6.1.3 Extracting OSP Messages From Stream Connections

Regardless of whether OSP is using a stream or RMTP transport connection, it always sends and receives data in structured chunks called messages. If the stream transport connection is through a true byte-oriented streaming transport protocol like TCP, the OSP message boundries will not be preserved in the stream. RMTP connections do not have this behavior. Some ordered-reliable protocols that are serving as stream transports for OSP will not have this behavior. SPX is one example.

6.1.3.1 Reassembling Messages

OSP must be able to reassemble its messages from the stream of bytes it receives through a stream transport connection. Since TCP is a streaming protocol it may arbitrarily break up a message from one side when sending to the other. For example the sending side may call xptSend() to send something that looks like this:

The TCP on the sending side sends it in 2 packets like this:

When the receiving side calls xptReceive() it gets this:

Then it calls xptReceive() again and gets this:

Because TCP may break up an OSP message in the middle of the OSP header, OSP may receive a partial OSP header from xptReceive().

6.1.3.2 Seperating Messages

OSP must also be able to fragment the stream of bytes it receives through a stream transport connection into messages. Since TCP is a streaming protocol it may arbitrarily combine multiple messages from one side when sending to the other. For example the sending side may call xptSend() to send something that looks like this:

Then the sending side may call xptSend() to send something that looks like this:

The TCP on the sending side sends both messages together in 1 packet like this:

When the receiving side calls xptReceive() it gets this:

6.1.4 Gateways

The architecture and theory of operation for HP-JetSend gateways is described in [2].

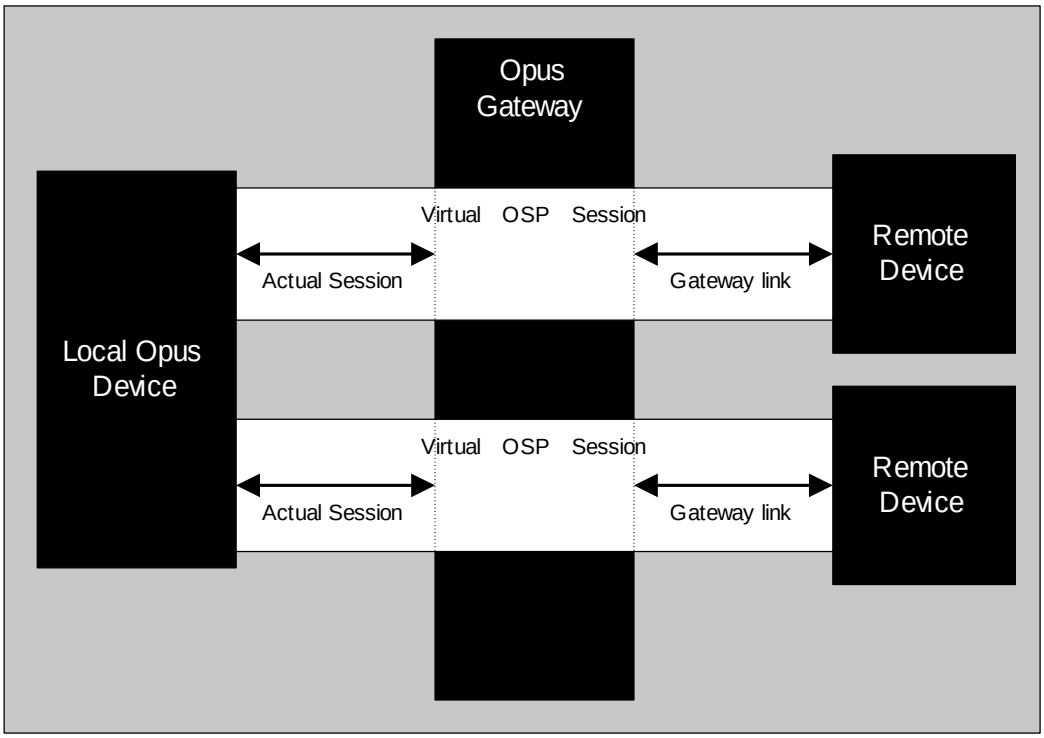


Figure 3

Although HP-JetSend does not allow multiple simultaneous sessions between the same two devices, HP-JetSend gateways are an exception. From the OIP perspective, there is only one session between itself and any other one device. However, an HP-JetSend gateway may act as a relay for multiple remote devices (e.g. FAX machines, E-Mail Servers). The source OSP establishes a session with the gateway for each device for which the gateway is a relay. Gateways create the requirement that OSP support multiple sessions with a single host and this leads to the requirement for the session fields in messages.

When OIP wants to establish a session with a remote device that requires an HP-JetSend gateway to access, OSP hides the existence of the gateway from OIP. OSP creates the session with the gateway instead and passes it the remote device's address in the SYN message's OMN Form Target Address field. From that point forward, when the gateway receives OIP data through any of the channels in the session, it relays the data to the remote device over the gateway link as illustrated in **Figure 3**.

When OIP wants to create a session to a second remote device that is accessed through the same gateway, OSP opens a second session with the gateway. An OSP implementation may (and likely would) share a single set of transport connections among the multiple sessions with the gateway.

6.1.4.1 Gateway Broadcasts

The OSP implementation should maintain an internal table that contains a mapping of gateway types to gateway addresses. The OSP API provides a function call – ospAddGateway() – that allows a user to enter the network address of a gateway.

When OSP receives a request from OIP to open a session with a device on a network that it does not support and to which it has no gateway listed in its table, it will go looking for one. It will broadcast a request to find out if there are any gateways that will relay messages to the device specified in the OIP request.

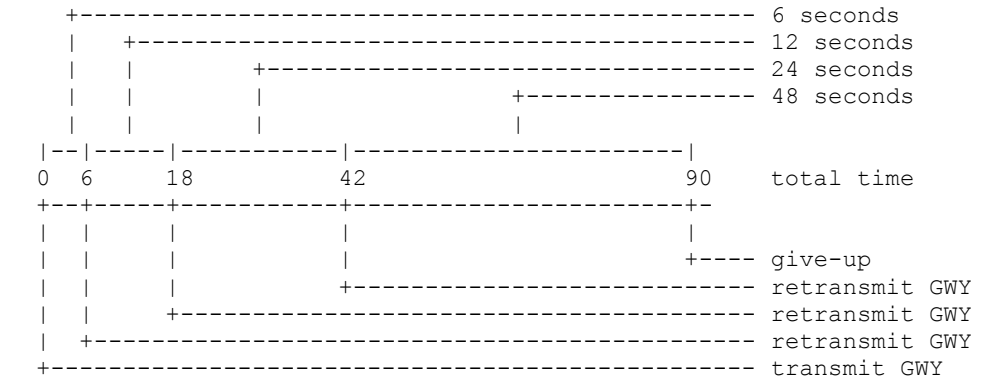
OSP broadcasts a gateway request by calling xptSendDatagram(). It then listens for a reply on the same local address through which it sent the broadcast by calling xptReceiveDatagram(). An HP-JetSend gateway listens for broadcasts on its well-known datagram transport port by calling xptReceiveDatagram(). When it receives a gateway broadcast, it looks to see if it can relay messages to the target address. If it can it replies to the broadcast using xptSendDatagram().

The gateway request is carried in a message with the GWY bit set (GWY message). It includes the address of the remote device for which a gateway is needed. The response to the GWY is a GWY message with the ACK bit also set (GACK message). It contains the network address of the gateway.

If OSP receives a GACK response to its GWY, it caches the gateway information in its internal table and establishes a session with the gateway. If it does not receive a GACK, it rejects the request from OIP for a session.

6.1.4.1.1 Gateway Broadcast Time-out

If a GACK does not arrive in a timely manner after OSP broadcasts a GWY message, OSP will use a timer to retransmit a GWY as described below. If OSP does not receive a response to a GWY within 90 seconds it gives up.



When OSP sends the first GWY, it sets a timer for 6 seconds. When the 6 seconds elapses and OSP has not received a GACK, it doubles the timer to 12 seconds and retransmits. Each time it retransmits it doubles the timer. When the total elapsed time reaches 90 seconds, OSP gives up. It will at this time reply to OIP that it cannot set up a session to the remote device.

6.2 OSP Messages and Formats

6.2.1 OSP Header Format

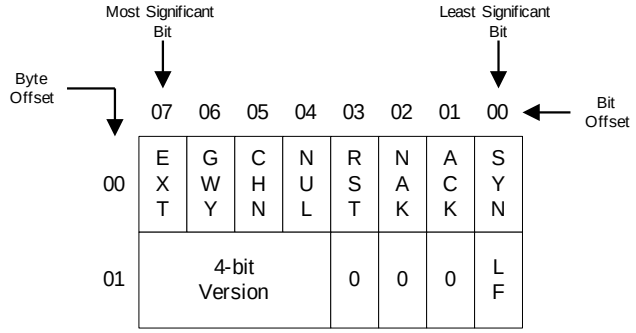


Figure 4

Every OSP message is prefaced with an OSP header. The format of the header is shown in **Figure 4**. Octet (Byte) offsets from the beginning of the message are numbered from top to bottom. Bit positions are numbered from right to left beginning with the least significant bit and ending with the most significant bit. The numerical value of a set bit is two raised to the power of the bit position.

All multi-byte words in a segment are transmitted across the network in network byte order – that is in big-endian form.

6.2.2 OSP Header Fields

6.2.2.1 Control Bits

This 12-bit field occupies the first octet and the first half of the second octet of the header. It is bit encoded with the following bits currently defined:

0) **SYN** Establish session or channel.
1) **ACK** Accept Session or channel.
2) **NAK** Reject session or channel.
3) **RST** Close the session or channel.
4) **NUL** Keep-alive.
5) **CHN** The message is directed to a specific channel.
6) **GWY** The message is a gateway broadcast or reply.
7) **EXT** Extended Header. Byte 3 contains additional control bits.
8) **LF** Fragment flag. If it is 0, this is not the last fragment. If it is 1, this is the last fragment.
9) Unused - set to 0.
10) Unused - set to 0.
11) Unused - set to 0.

The control bits are used in combinations to create these message types (The EXT bit is 0 in all these combinations):

SYN	ACK	NAK	RST	NUL	CHN	Name	Meaning
1	0	0	0	0	0	SYN	Request for session.
1	**1**	0	0	0	0	ASYN	Accept session request.
1	0	**1**	0	0	0	NSYN	Reject session request.
0	0	0	**1**	0	0	RST	Close session.
0	0	0	0	**1**	0	NUL	Keep-alive.

0	0	0	0	0	**1**	CHN	Data for a channel.
1	0	0	0	0	**1**	CSYN	Request for channel.
1	**1**	0	0	0	**1**	CASYN	Accept channel request.
1	0	**1**	0	0	**1**	CNSYN	Reject channel request.
0	0	0	**1**	0	**1**	CRST	Close channel.

ACK	CHN	Name	Meaning
0	**1**	GWY	Gateway broadcast.
1	**1**	GACK	Gateway broadcast reply.

6.2.2.2 Version Number

This 4-bit field occupies last half of the second octet (bits 4-7). It contains the version number of the protocol described by this document. Current value is zero (0).

6.2.3 Extended Header Format

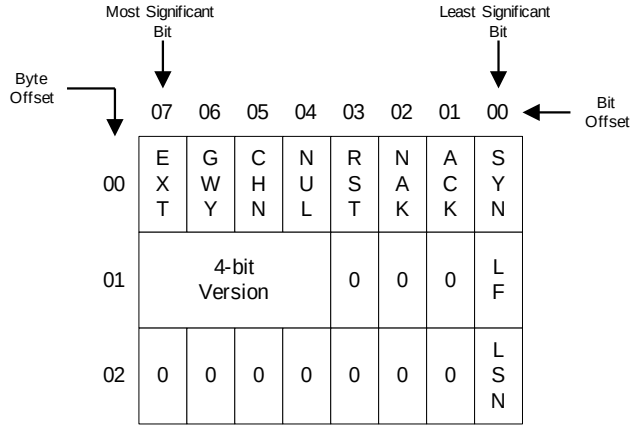


Figure 5

If the EXT bit is set in the header, the header contain an additional octet of control bits. The format of the header is shown in **Figure 5**.

6.2.4 Extended Header Fields

6.2.4.1 Extended Control Bits

This 8-bit field occupies the third octet of the header. It is bit encoded with the following bits currently defined:

0) **LSN** The sending session has started or stopped listening for channels.
1) Unused. Set to 0.
2) Unused. Set to 0.
3) Unused. Set to 0.
4) Unused. Set to 0.
5) Unused. Set to 0.
6) Unused. Set to 0.
7) Unused. Set to 0.

6.2.5 SYN Message (0x01)

OSP uses the SYN message to request a session. The SYN is sent as a separate message and does not include any data.

6.2.5.1 Message Format

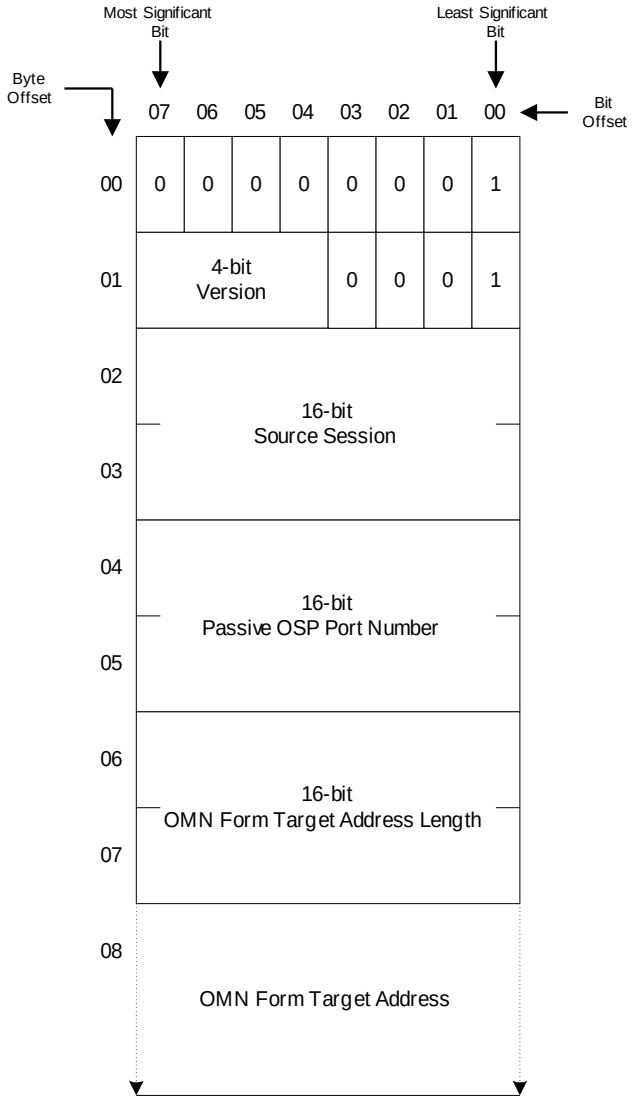


Figure 6

6.2.5.2 Message Fields

6.2.5.2.1 Source Session

The session from which the sending side of OSP sent this message.

6.2.5.2.2 Passive OSP Port Number

The Passive OSP port number field contains the OSP port number of the HP-JETSEND device in the remote host with which OSP should establish a session. This value is variously refered to in the

HP-JETSEND documents as the "sub-address", "device-number", and "id-string".

6.2.5.2.3 OMN Form Target Address Length

The length in octets (bytes) of the OMN Form Target Address field including the terminating zero byte. The value in this field is 0 unless OSP is requesting a session with a gateway.

6.2.5.2.4 OMN Form Target Address

The address of a remote device in HP-JetSend Multi-Network (OMN) form [3]. This field is empty unless OSP is requesting a session with a gateway. See section 6.1.4 for a description of how OSP uses this field. The OMN form address is stored in this field as a zero terminated ASCII string.

6.2.6 ASYN Message (0x03)

OSP uses the ASYN message to acknowledge a SYN message and accept the SYN's session request. The ASYN is sent as a separate message and does not include any data.

6.2.6.1 Message Format

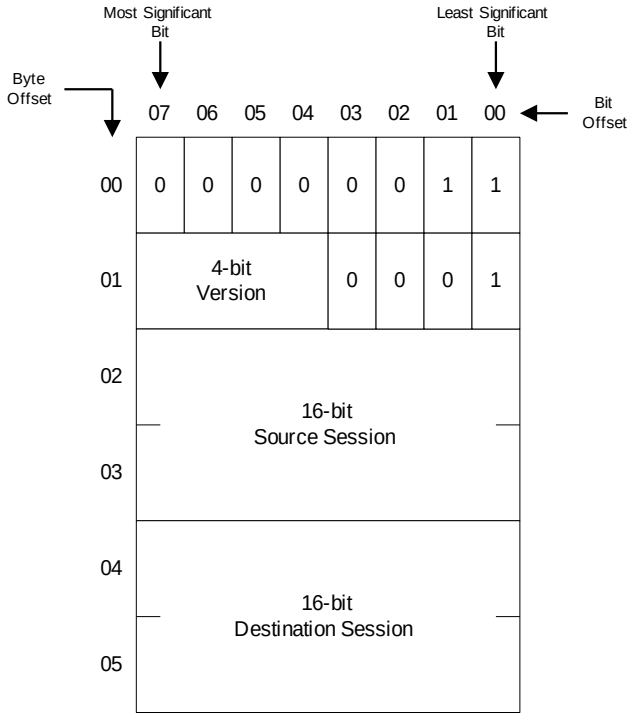


Figure 7

6.2.6.2 Message Fields

6.2.6.2.1 Source Session

The session from which the sending side of OSP sent this message.

6.2.6.2.2 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.7 NSYN Message (0x05)

OSP uses the NSYN message to acknowledge a SYN message and reject the SYN's session request. The NSYN is sent as a separate message and does not include any data.

6.2.7.1 Message Format

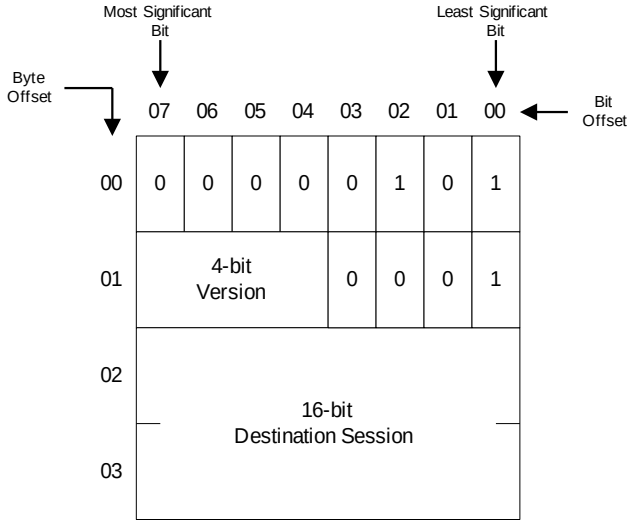


Figure 8

6.2.7.2 Message Fields

6.2.7.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.8 RST Message (0x08)

OSP uses the RST message close a session. Upon receipt of an RST message, the receiverer must stop sending and must abort any unserviced requests. The RST is sent as a separate message and does not include any data.

6.2.8.1 Message Format

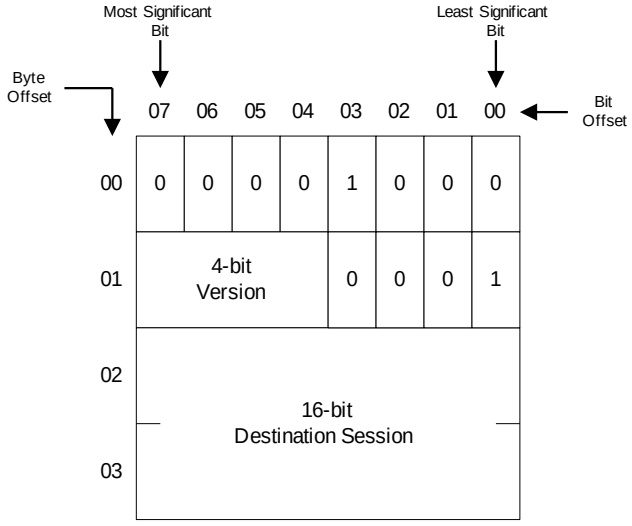


Figure 9

6.2.8.2 Message Fields

6.2.8.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.9 NUL Message (0x10)

OSP uses the NUL message to detect a half-open session. Its use is described in 6.1.1.5.

6.2.9.1 Message Format

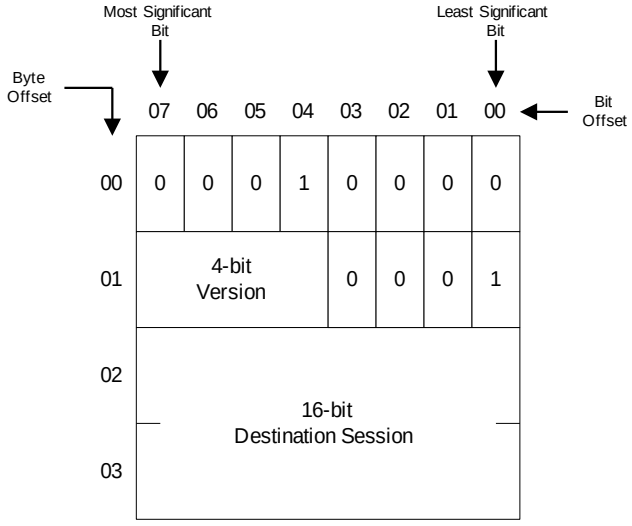


Figure 10

6.2.9.2 Message Fields

6.2.9.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.10 CHN Message (0x20)

6.2.10.1 Message Format

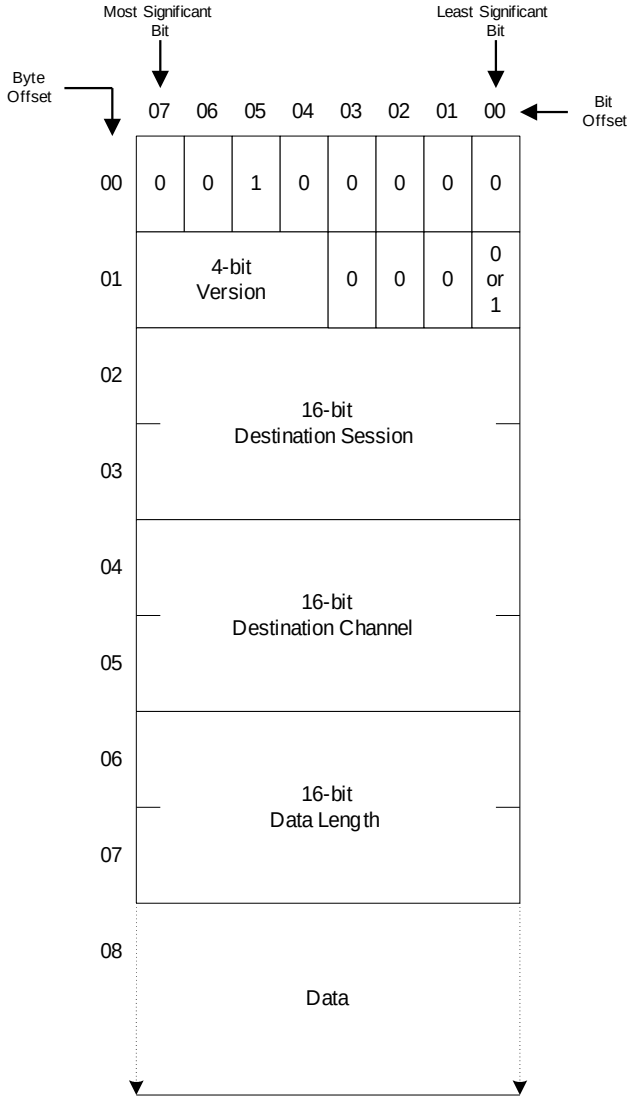


Figure 11

6.2.10.2 Message Fields

6.2.10.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.10.2.2 Destination Channel

The channel to which the receiving side of OSP should route this message.

6.2.10.2.3 Data Length

The length in octets (bytes) of the Data field.

6.2.10.2.4 Data

A fragment of the OIP data that OSP has been told to send to a remote host.

6.2.11 CSYN Message (0x21)

OSP uses the CSYN message to request a channel. The CSYN is sent as a separate message and does not include any data.

6.2.11.1 Message Format

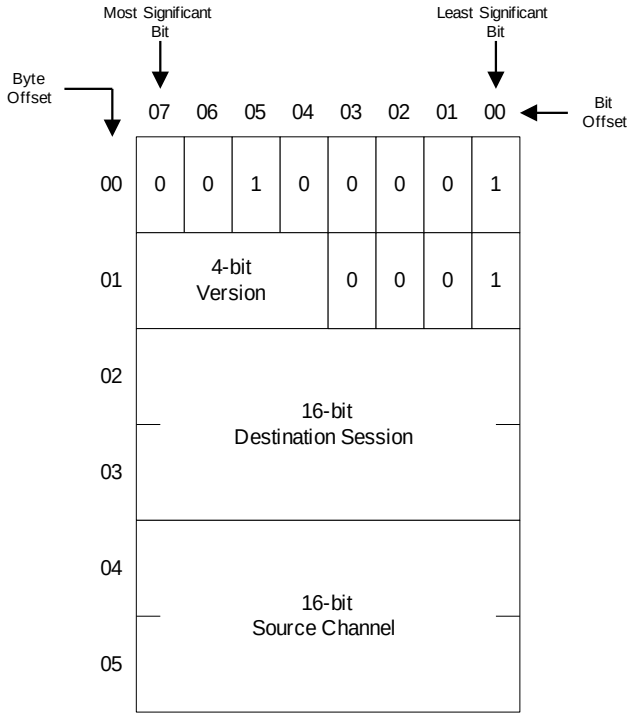


Figure 12

6.2.11.2 Message Fields

6.2.11.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.11.2.2 Source Channel

The channel from which the sending side of OSP sent this message.

6.2.12 CASYN Message (0x23)

OSP uses the CASYN message to acknowledge a CSYN message and accept the CSYN's channel request. The CASYN is sent as a separate message and does not include any data.

6.2.12.1 Message Format

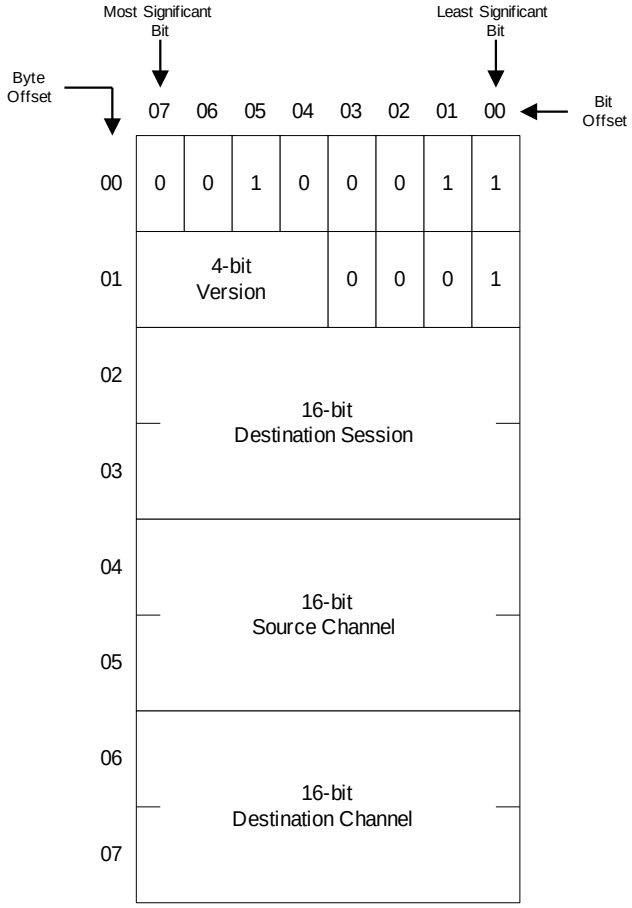


Figure 13

6.2.12.2 Message Fields

6.2.12.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.12.2.2 Source Channel

The channel from which the sending side of OSP sent this message.

6.2.12.2.3 Destination Channel

The channel to which the receiving side of OSP should route this message.

6.2.13 CNSYN Message (0x25)

OSP uses the CNSYN message to acknowledge a CSYN message and reject the CSYN's channel request. The CNSYN is sent as a separate message and does not include any data.

6.2.13.1 Message Format

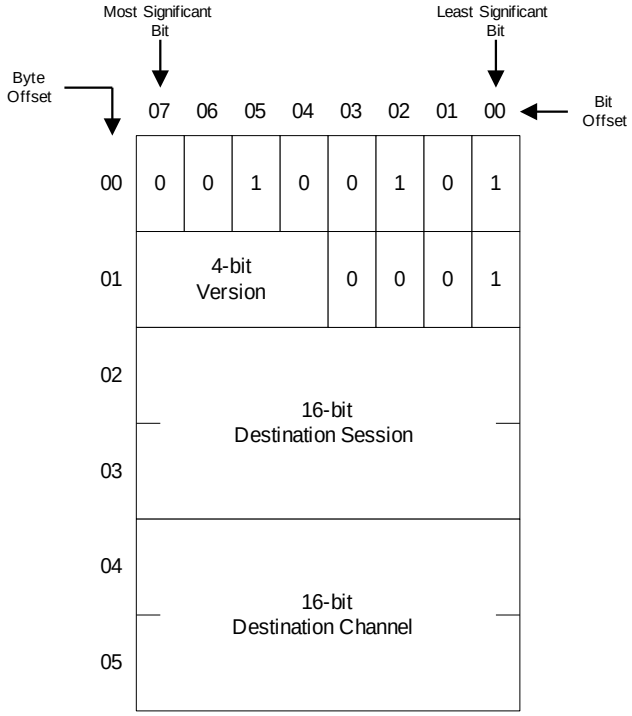


Figure 14

6.2.13.2 Message Fields

6.2.13.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.13.2.2 Destination Channel

The channel to which the receiving side of OSP should route this message.

6.2.14 CRST Message (0x28)

OSP uses the CRST message to close a channel. Upon receipt of an CRST message, the receiver must stop sending and must abort any unserviced requests. The CRST is sent as a separate message and does not include any data.

6.2.14.1 Message Format

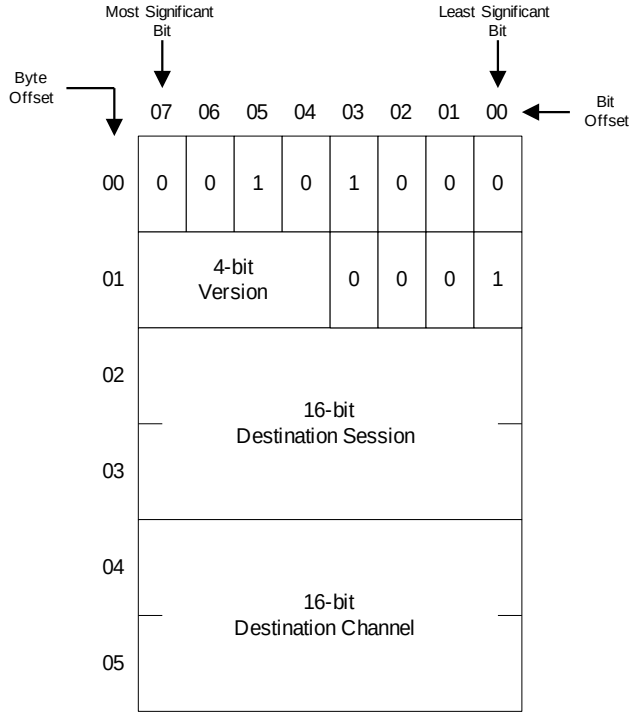


Figure 15

6.2.14.2 Message Fields

6.2.14.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.14.2.2 Destination Channel

The channel to which the receiving side of OSP should route this message.

6.2.15 CEXTLSN Message (0xA0)

OSP uses the CEXTLSN message to tell its peer that it has either started or has stopped listening for channels within a session. The message specifies which type of channel – MESSAGE or STREAM – that OSP is listening or not listening for. When OSP receives a CEXTLSN from its peer indicating that the peer is listening for channels, OSP may try to actively establish new channels of that type with its peer. When OSP receives a CEXTLSN from its peer indicating that the peer is no longer listening for channels, OSP should not try to actively establish new channels of that type with its peer. OSP can set both the MSG and STR bits in one CEXTLSN message to indicate that it has started or stopped listening for channels of both types.

6.2.15.1 Message Format

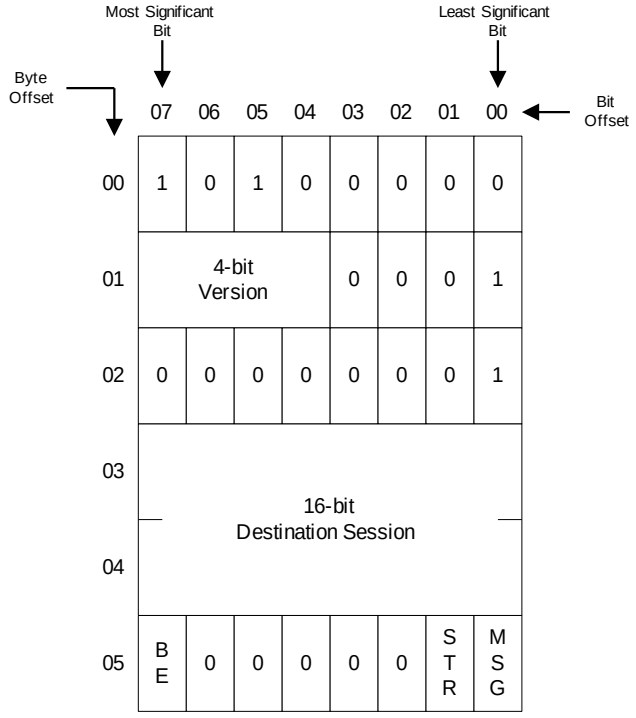


Figure 16

6.2.15.2 Message Fields

6.2.15.2.1 Destination Session

The session to which the receiving side of OSP should route this message.

6.2.15.2.2 Message data bits.

This 8-bit field occupies the sixth octet of the message. It is bit encoded with the following bits currently defined:

0) **MSG** OSP has started or stopped listening for MESSAGE channels.
1) **STR** OSP has started or stopped listening for STREAM channels.
2) Unused. Set to 0.
3) Unused. Set to 0.
4) Unused. Set to 0.
5) Unused. Set to 0.
6) Unused. Set to 0.
7) **BE** Begin/End bit. If this bit is set (1), OSP has started listening for MESSAGE and/or STREAM channels. If this bit is not set (0), OSP has stopped listening for MESSAGE and/or STREAM channels.

6.2.16 GWY Message (0x40)

OSP uses the GWY message to locate a gateway. The GWY is sent as a separate message and does not include any data.

6.2.16.1 Message Format

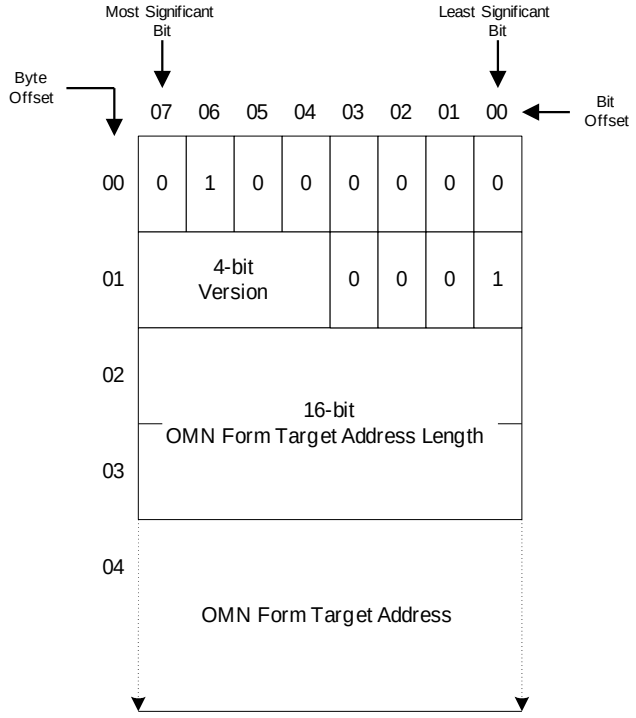


Figure 17

6.2.16.2 Message Fields

6.2.16.2.1 OMN Form Target Address Length

The length in octets (bytes) of the OMN Form Target Address field including the terminating zero byte.

6.2.16.2.2 OMN Form Target Address

The address in HP-JetSend Multi-Network (OMN) form of the remote device for which OSP is requesting a gateway [3]. The OMN form address is stored in this field as a zero terminated ASCII string.

6.2.17 GACK Message (0x42)

OSP uses the GACK message to respond to a GWY message. The GACK is sent as a separate message and does not include any data.

6.2.17.1 Message Format

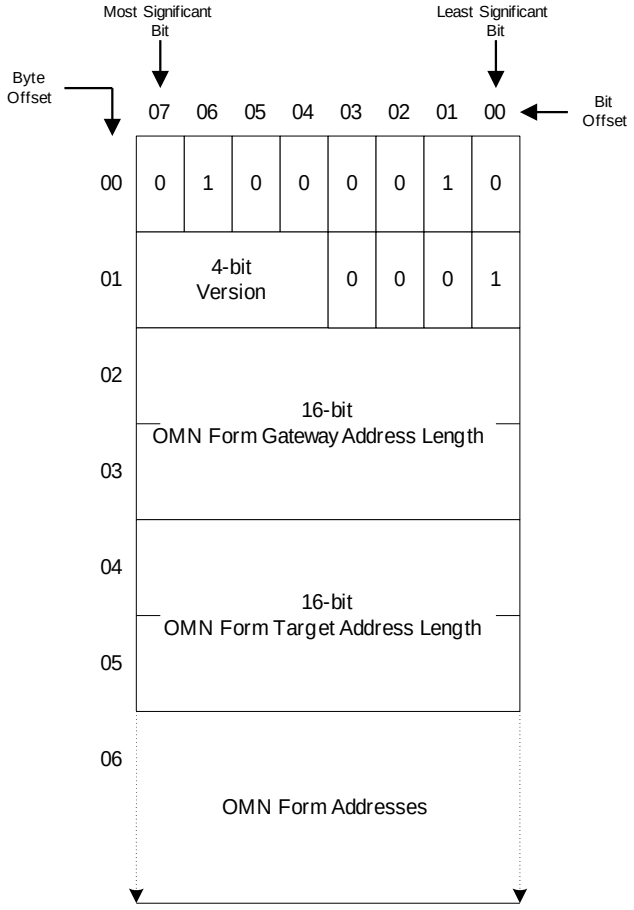


Figure 18

6.2.17.2 Message Fields

6.2.17.2.1 OMN Form Gateway Address Length

The length in octets (bytes) of the responding gateway's address in the OMN Form Addresses field including the terminating zero byte. The gateway's address is the first OMN form address in the OMN Form Addresses field.

6.2.17.2.2 OMN Form Target Address Length

The length in octets (bytes) of the remote device's address in the OMN Form Addresses field including the terminating zero byte. The remote device's address is copied from the GWY message. It is returned in the GACK so that OSP can match up GACK messages with GWY messages that it has broadcast. The remote device's address is the second OMN form address in the OMN Form Addresses field.

6.2.17.2.3 OMN Form Addresses

The addresses in HP-JetSend Multi-Network (OMN) form of the responding gateway and the remote device for which OSP is requesting a gateway [3]. OMN form addresses are stored in this field as zero terminated ASCII strings.