# Intel® AMT Port Forwarding Protocol Reference Manual

Release 1.0.5 June 2008

# Table of Contents

# 1 Introduction

The Intel® **A**MT **p**ort **f**orwarding (APF) protocol provides TCP and UDP connection multiplexing over a single reliable transport session, typically a TLS session [TLS] or a reliable hardware bus interface, for example, the Intel® Management Engine Interface (Intel® MEI, or MEI), also known as HECI. This mechanism is useful for enabling client-server communication, where the two peers are located on different intranets.

The APF protocol assumes that any aspects of confidentiality and server authentication are handled by the underlying transport session. Also, some aspects of the authentication may be provided by communication sessions running on top of the Intel® AMT port forwarding protocol.

## 1.1 Overall Protocol Architecture

The APF protocol is based on components adapted from the Secure Shell Protocol [SSH], specifically the SSH Transport Layer Protocol [SSH-Trans], which provides the notion of a "service". The specification supports two services — a client authentication service, derived from the SSH Authentication Protocol [SSH-Auth], and a port forwarding service, derived from the SSH Connection Protocol [SSH-Connect]. The APF protocol aligns as much as possible to the message syntax and state machine defined in the relevant SSH protocols.

## 1.2 Services

The client sends a service request for user authentication, once the underlying session has been established (MEI or TLS). A second service request for port forwarding should be sent only after user authentication is complete. A client may skip the authentication service when client authentication is not required if, for example, TLS mutual authentication is to be used.

## 1.3 Client authentication

The Client authentication service is derived from [SSH-Auth]. The APF supports only two authentication modes — "none" and "password".

The "none" method MAY be sent by the client. The server MUST always reject this request, unless the client is to be granted access without any authentication, in which case the server MUST accept this request. The main purpose of sending this request is to get the list of supported methods from the server.

In the "password" method, the password is sent as clear-text to the server. Note that the use of TLS with server side authentication, guarantees the confidentiality of the password information.

The server SHOULD have a timeout for authentication and disconnect if the authentication has not been accepted within the timeout period. Additionally, the implementation SHOULD limit the number of failed authentication attempts a client may perform in a single session. If this threshold is exceeded, the server SHOULD disconnect.

## 1.4 Port Forwarding Mechanism

Both parties deploy a port forwarding agent. These two agents are symmetric in nature but an implementation can limit support based on actual use on each end of the connection. The agents can route multiple TCP and UDP connections received from client entities located in the intranet to server entities located at the peer's intranet via the peer agent.

This protocol does not specify how routing is established from a TCP or UDP client to a port forwarding agent. Implementers can choose from a variety of standard protocols such as SOCKSv5 [SOCKSv5] or HTTP Proxy [HTTP].

As stated above, the port forwarding mechanism is aligned to a great extent with [SSH-Connect]. [SSH-Connect] defines a mechanism that provides port forwarding of multiple TCP connections over a single SSH session. [SSH-Connect] is tightly coupled to SSH [SSH] and cannot be applied to alternative secure and reliable protocols such as TLS.

### 1.4.1 Channel Mechanism

All forwarded connections are channels; either side may open a channel. Multiple channels are multiplexed into a single connection.

Channels are identified by numbers at each end. The number referring to a channel may be different on each side. Requests to open a channel contain the sender's channel number. Any other channel-related messages contain the recipient's number for the channel.

Channels are flow-controlled. No data may be sent to a channel until a message is received to indicate that window space is available.

# 2 References


[SSH] IETF RFC 4251: The Secure Shell (SSH) Protocol Architecture

[SSH-Connect] IETF RFC 4254: The Secure Shell (SSH) Connection Protocol

[SSH-Auth] IETF RFC 4252: The Secure Shell (SSH) Authentication Protocol

[SSH-Trans] IETF RFC 4253: The Secure Shell (SSH) Transport Layer Protocol

[TLS] IETF RFC 2246: The TLS Protocol Version 1.0

[HTTP] IETF RFC 2616: Hypertext Transfer Protocol -- HTTP/1.1

[SOCKSv5] IETF RFC 1928: SOCKS Protocol Version 5


# 3 Message Definitions

The following is a summary of protocol messages and their associated message number.

```
APF_DISCONNECT                  1
APF_SERVICE_REQUEST             5
APF_SERVICE_ACCEPT              6
APF_USERAUTH_REQUEST            50
```

```
APF_USERAUTH_FAILURE                51
APF_USERAUTH_SUCCESS                52
APF_GLOBAL_REQUEST                  80
APF_REQUEST_SUCCESS                 81
APF_REQUEST_FAILURE                 82
APF_CHANNEL_OPEN                    90
APF_CHANNEL_OPEN_CONFIRMATION       91
APF_CHANNEL_OPEN_FAILURE            92
APF_CHANNEL_WINDOW_ADJUST           93
APF_CHANNEL_DATA                    94
APF_CHANNEL_CLOSE                   97
APF_PROTOCOLVERSION                 192
APF_KEEPALIVE_REQUEST               208
APF_KEEPALIVE_REPLY                 209
APF_KEEPALIVE_OPTIONS_REQUEST       210
APF_KEEPALIVE_OPTIONS_REPLY         211
```

## 3.1 Transport Layer Messages

The following messages are adopted from the SSH Transport Layer Protocol [SSH-Trans].

### 3.1.1 APF_DISCONNECT

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_DISCONNECT |
| 1 | 4 | Reason code |
| 5 | 2 | Reserved = 0 |

*Reason Code* — A Reason code for the disconnection event. See table below for possible reasons

*Reserved* — Reserved must be set to 0

This message causes immediate termination of the connection. All implementations MUST be able to process this message; they SHOULD be able to send this message. The sender MUST NOT send or receive any data after this message, and the recipient MUST NOT accept any data after receiving this message.

| Symbol | Code |
|---|---|
| APF_DISCONNECT_HOST_NOT_ALLOWED_TO_CONNECT | 1 |
| APF_DISCONNECT_PROTOCOL_ERROR | 2 |
| APF_DISCONNECT_KEY_EXCHANGE_FAILED | 3 |
| APF_DISCONNECT_RESERVED | 4 |
| APF_DISCONNECT_MAC_ERROR | 5 |
| APF_DISCONNECT_COMPRESSION_ERROR | 6 |
| APF_DISCONNECT_SERVICE_NOT_AVAILABLE | 7 |
| APF_DISCONNECT_PROTOCOL_VERSION_NOT_SUPPORTED | 8 |

| Symbol | Code |
|---|---|
| APF_DISCONNECT_HOST_KEY_NOT_VERIFIABLE | 9 |
| APF_DISCONNECT_CONNECTION_LOST | 10 |
| APF_DISCONNECT_BY_APPLICATION | 11 |
| APF_DISCONNECT_TOO_MANY_CONNECTIONS | 12 |
| APF_DISCONNECT_AUTH_CANCELLED_BY_USER | 13 |
| APF_DISCONNECT_NO_MORE_AUTH_METHODS_AVAILABLE | 14 |
| APF_DISCONNECT_INVALID_CREDENTIALS | 15 |
| APF_DISCONNECT_CONNECTION_TIMED_OUT | 16 |
| APF_DISCONNECT_BY_POLICY | 17 |
| APF_DISCONNECT_TEMPORARILY_UNAVAILABLE | 18 |

### 3.1.2 APF_SERVICE_REQUEST

The service is identified by a name. If the server rejects the service request, it SHOULD send an appropriate APF_DISCONNECT message and MUST disconnect. If the server supports the service (and permits the client to use it), it MUST respond with an APF_SERVICE_ACCEPT message indicating the corresponding service.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_SERVICE_REQUEST |
| 1 | 4 | Service Name string length (N) |
| 5 | N | Service Name string |

*`Service Name`* The name of the service being requested. The APF protocol supports only the following service names:
pfwd@amt.intel.com
auth@amt.intel.com

### 3.1.3 APF_SERVICE_ACCEPT

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_SERVICE_ACCEPT |
| 1 | 4 | Service Name string length (N) |
| 5 | N | Service Name string |

*`Service Name`* The name of the service that was sent in the previous APF_SERVICE_REQUEST message.

### 3.1.4 APF_ProtocolVersion

Protocol versioning messages must be the first messages exchanged once a session is open. A party MUST first send and receive this message before any additional messages are sent. A party MUST send this message only once.

The negotiated version is the lower version value sent by the two peers; if the peer sending the higher version ID does not support the negotiated version it MUST terminate the session.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_PROTOCOLVERSION |
| 1 | 4 | Major version |
| 5 | 4 | Minor version |
| 9 | 4 | Reserved |
| 13 | 16 | APF System ID (UUID) |
| 29 | 64 | Reserved |

*Major version* The major number of the protocol version supported by the sender.

*Minor version* The minor number of the protocol version supported by the sender.

*APF System ID* System ID

### 3.1.5 Keepalive messages

#### 3.1.5.1 APF_Keepalive_Request

Anytime a party is connected it MAY send an APF_Keepalive_Request message.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_KEEPALIVE_REQUEST |
| 1 | 4 | Keepalive cookie |

*Keepalive cookie* Cookie to be sent back in the reply.

#### 3.1.5.2 APF_Keepalive_Reply

If a peer receives an APF_Keepalive_Request and no other output is pending to be sent, it MUST send an APF_Keepalive_Reply.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_KEEPALIVE_REPLY |
| 1 | 4 | Keepalive cookie |

*Keepalive cookie* Cookie received in the previous APF_KeepaliveRequest.

#### 3.1.5.3 APF_Keepalive_Options_Request

Anytime a party is connected it MAY send an APF_KeepaliveOptions message to specify how it would like the other party to handle keepalives. The receiving party must decide whether or not to accept the requested values.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_KEEPALIVE_OPTIONS_REQUEST |
| 1 | 4 | Keepalive interval |
| 5 | 4 | Read timeout |

*Keepalive interval* If non-zero and no other data is sent for this many seconds, send an APF_KeepaliveRequest.

*Read timeout* If non-zero and no data is received for this many seconds, the connection is to be considered broken.

#### 3.1.5.4 APF_Keepalive_Options_Reply

If a peer receives an APF_Keepalive_Options, it MUST send an APF_Keepalive_Options_Reply specifying the values it has decided on using.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_KEEPALIVE_OPTIONS_REPLY |
| 1 | 4 | Keepalive interval |
| 5 | 4 | Read timeout |

*Keepalive interval* If non-zero and no other data is sent for this many seconds, the peer will send an APF_KeepaliveRequest.

*Read timeout* If non-zero and no data is received for this many seconds, the peer will consider the connection to be broken.

## 3.2 Authentication Messages

### 3.2.1 APF_USERAUTH_REQUEST

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_USERAUTH_REQUEST |
| 1 | 4 | Username String length (U) |
| 5 | U | Username string |
| U+5 | 4 | Service Name String Length (S) |
| U+9 | S | Service Name = "pfwd@amt.intel.com" |
| U+S+9 | 4 | Method Name String Length (MN) |
| U++S+13 | MN | Method Name String |

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| … | | Method specific fields |

*Username String* — user name in ASCII encoding; the maximum allowed string size is 64 bytes.

*Service Name* — The Service Name to authorize. The only supported service name is "pfwd@amt.intel.com".

*Method Name String* — The authentication method to use; the only defined methods are "none" and "password".

#### 3.2.1.1 "none" authentication method

A client may request a list of authentication 'method name' values by using the "none" authentication 'method name'. If no authentication is needed for the user, the server MUST return APF_USERAUTH_SUCCESS. Otherwise, the server MUST return APF_USERAUTH_FAILURE and MAY return with it a list of valid authentication methods in its 'Method name list' field.

#### 3.2.1.2 "password" authentication method

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| U+S+MN + 13 | 1 | Reserved = 0 |
| U+S+MN+14 | 4 | Password length(P) |
| U+S+MN+18 | P | Password string |

*Reserved* — Reserved MUST be set to 0.

*Password String* — Plain text password in ASCII format, the maximum allowed password size is 64 bytes.

### 3.2.2 APF_USERAUTH_FAILURE

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_USERAUTH_FAILURE |
| 1 | 4 | Method name-list length(ML) |
| ML+5 | ML | Method name-list ="password" |
| ML+6 | 1 | Reserved = 0 |

*Method Name list* — A comma-separated string of authentication methods supported by the server, in ASCII. In the APF protocol, this field must be set to "password". Refer to [SSH] for the structure of name-list.

*Reserved* — Reserved MUST be set to 0.

### 3.2.3 APF_USERAUTH_SUCCESS

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_USERAUTH_SUCCESS |

## 3.3 Port Forwarding Messages

The APF protocol is based on three types of messages: **global messages**, **authentication messages**, and **channel messages**.

The reply messages do not include request type identifiers. To make it possible for the originator of a request to identify which request each reply refers to, it is required that replies to global messages be sent in the same order as the corresponding request messages. For channel requests, replies that relate to the same channel must also be replied to in the order received. However, channel requests for distinct channels may be replied to out-of-order.

### 3.3.1 Global Messages

There are several kinds of requests that affect the state of the remote end globally, independent of any channels. An example is a request to start TCP/IP forwarding for a specific port. Note that both the client and server may send global requests at any time, and the receiver must respond appropriately.

In general, the reply messages do not include request type identifiers. To make it possible for the originator of a request to identify to which request each reply refers, it is REQUIRED that replies to APF_GLOBAL_REQUESTS MUST be sent in the same order as the corresponding request messages.

All global request messages contain the following header.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_GLOBAL_REQUEST |
| 1 | 4 | Request string length (N) |
| 5 | N | Request string |
| 5 + N | 1 | Want reply |

*`Request string length (N)`* Number of bytes in the following string. Empty string is represented by N=0.

*`Request string`* The request string. The string should not be terminated with a NULL character.

*`Want reply`* Indicates if a reply is needed. 0=No-Reply, 1=Reply.

#### 3.3.1.1 APF_TcpForwardRequest

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | H | Global request header. Request string = "tcpip-forward". Want reply = 1. |

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| H | 4 | Address-to-bind string length (M) |
| H + 4 | M | Address-to-bind string |
| H + 4 + M | 4 | Port number to bind |

*Address-to-bind string* The IP address or hostname on which connections for forwarding are to be accepted.

Some strings have special-case semantics:

"0.0.0.0" refers to all IPv4 addresses.

"::" refers to all IPv6 addresses.

"127.0.0.1" and "::1" refer to the loopback interfaces for IPv4 and IPv6, respectively.

*Port number to bind* The TCP port on which connections for forwarding are to be cancelled. Only non-zero values are supported.

#### 3.3.1.2 APF_TcpForwardReply

If the request is successful, the following message will be returned:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_REQUEST_SUCCESS |
| 1 | 4 | Port bound |

*Port bound* The TCP port that was bound on the server.

If the request failed, the following message will be returned:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_REQUEST_FAILURE |

#### 3.3.1.3 APF_TcpForwardCancelRequest

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | H | Global request header. Request string = "cancel-tcpip-forward".<br>Want reply = 1. |
| H | 4 | Address-to-bind string length (M) |
| H + 4 | M | Address-to-bind string |
| H + 4 + M | 4 | Port number to bind |

*Address-to-bind string* The IP address or hostname on which connections for forwarding are to be canceled.

Some strings have special-case semantics:

"0.0.0.0" refers to all IPv4 addresses.

"::" refers to all IPv6 addresses.

"127.0.0.1" and "::1" refer to the loopback interfaces for IPv4 and IPv6, respectively.

*Port number to bind* The TCP port on which connections for forwarding are to be canceled. Only non-zero values are supported.

Using APF_TcpForwardRequest with a host-name and APF_TcpForwardCancelRequest with an IP address or vice versa is invalid.

### 3.3.1.4 APF_TcpForwardCancelReply

If the request is successful, the following message will be returned:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_REQUEST_SUCCESS |

If the request failed, the following message will be returned:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_REQUEST_FAILURE |

### 3.3.1.5 APF_UdpSendTo

UDP support is limited to unidirectional datagram forwarding. Recipient may silently drop the datagram. Datagram size must be < 64KB.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | H | Global request header. Request string = "udp-send-to@amt.intel.com".<br>Want reply = 0. |
| H | 4 | Host-to-connect string length (M) |
| H + 4 | M | Host-to-connect string |
| H + M + 4 | 4 | Port number to connect |
| H + M + 8 | 4 | Originator IP address string length (L) |
| H + M + 12 | L | Originator IP address string |
| H + M + L + 12 | 4 | Originator port |
| H + M + L + 16 | 4 | Data length (D) |
| H + M + L + 20 | D | Data |

*Host-to-connect string* The IP address or hostname to send the datagram to.

*Port number to connect*
: The UDP port number to send the datagram to.

*Originator IP address*
: The IP address of the sender as will appear in the UDP packet.

*Originator port*
: The UDP port of the sender as will appear in the UDP packet. The value ‘0’ means don’t care.

*Data length*
: Length (in bytes) of the following data. The length must be less than 64KB.

*Data*
: The data blob containing the UDP packet.

## 3.3.2 Channel Messages

### 3.3.2.1 APF_ChannelOpenForwardedRequest

When a connection comes to a port for which remote forwarding has been requested, a channel is opened to forward the port to the other side.

Implementations MUST reject these messages unless there was previously request for remote TCP/IP port forwarding with the given port number.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_OPEN |
| 1 | 4 | Channel type string length (T) |
| 5 | T | Channel type string = "forwarded-tcpip" |
| T + 5 | 4 | Sender channel |
| T + 9 | 4 | Initial window size |
| T + 13 | 4 | Reserved (0xFFFFFFFF) |
| T + 17 | 4 | Connected address string length (C) |
| T + 21 | C | Connected address string |
| T + C + 21 | 4 | Connected port |
| T + C + 25 | 4 | Originator IP address string length (O) |
| T + C + 29 | O | Originator IP address string |
| T + C + O + 29 | 4 | Originator port |

*Sender channel*
: The channel number assigned by the sender.

*Initial window size*
: Number of bytes in the window.

*Connected address string*
: Address that was connected.

*Connected port*
: The TCP port that was connected.

*Originator address string* IP address of the platform where the connection request originates.

*Originator port* TCP port on the platform where the connection request originates.

### 3.3.2.2 APF_ChannelOpenDirectRequest

When a connection comes to a locally forwarded TCP/IP port, the following packet is sent to the other side. Note that these messages may also be sent for ports for which no forwarding has been explicitly requested. The receiving side must decide whether to allow the forwarding. Note that the client must send an APF_SERVICE_REQUEST prior to sending service–related commands such as APF_ChannelOpenDirectRequest.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_OPEN |
| 1 | 4 | Channel type string length (T) |
| 5 | T | Channel type string = "direct-tcpip" |
| T + 5 | 4 | Sender channel |
| T + 9 | 4 | Initial window size |
| T + 13 | 4 | Reserved (0xFFFFFFFF) |
| T + 17 | 4 | Host-to-connect string length (C) |
| T + 21 | C | Host-to-connect string |
| T + C + 21 | 4 | Port to connect |
| T + C + 25 | 4 | Originator IP address string length (O) |
| T + C + 29 | O | Originator IP address string |
| T + C + O + 29 | 4 | Originator port |

*Sender channel* The channel number assigned by the sender.

*Initial window size* Number of bytes in the window.

*Host-to-connect string* IP address or hostname to connect the channel to.

*Port to connect* Port to connect the channel to.

*Originator address string* IP address of the platform where the connection request originates.

*Originator port* TCP port on the platform where the connection request originates.

### 3.3.2.3 APF_ChannelOpenReply

If the channel can be opened, the following reply message should be sent:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_OPEN_CONFIRMATION |
| 1 | 4 | Recipient channel |
| 5 | 4 | Sender channel |
| 9 | 4 | Initial window size |
| 13 | 4 | Reserved (0xFFFFFFFF) |

*Recipient channel* The channel number given in the open request.

*Sender channel* The channel number assigned by the sender.

*Initial window size* Number of bytes in the window.

If the channel cannot be opened, the following message should be sent:

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_OPEN_FAILURE |
| 1 | 4 | Recipient channel |
| 5 | 4 | Reason code |
| 9 | 4 | Reserved (0x00000000) |
| 13 | 4 | Reserved (0x00000000) |

*Recipient channel* The channel number given in the open request.

*Reason code* Reason the channel could not be opened.

Valid reason codes:

1 = OPEN_ADMINISTRATIVELY_PROHIBITED

2 = OPEN_CONNECT_FAILED

3 = OPEN_UNKNOWN_CHANNEL_TYPE

4 = OPEN_RESOURCE_SHORTAGE

#### 3.3.2.4 APF_ChannelClose

When a party wishes to terminate a channel, it sends the APF_ChannelClose message. The receiving party must send back an APF_ChannelClose message unless it has already sent this message for the same channel. The channel is considered closed for a party when it has both sent and received APF_ChannelClose. The party may then reuse the channel number.

This message does not consume window space and can be sent even if no window space is available.

APF_ChannelWindowAdjust messages may still be sent after sending this message, until the APF_ChannelClose message has been both sent and received.

It is recommended that all data is delivered to the actual destination before this message is sent, if possible.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_CLOSE |
| 1 | 4 | Recipient channel |

*Recipient channel* The channel number of the recipient.

### 3.3.2.5 APF_ChannelData

This message is used to transfer data.

The maximum amount of data allowed is determined by the current window size. The window size is decremented by the amount of data sent. Both parties may ignore all extra data sent after the allowed window is empty.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_DATA |
| 1 | 4 | Recipient channel |
| 5 | 4 | Length of data (L) |
| 9 | L | Data |

*Recipient channel* The channel number of the recipient.

*Length of data* The length (in bytes) of the following data.

*Data* The data.

### 3.3.2.6 APF_ChannelWindowAdjust

The window size specifies how many bytes the other party can send before it must wait for the window to be adjusted. Both parties use the following message to adjust the window.

After receiving this message, the recipient increments the window size by the Bytes to Add value. Implementations must correctly handle total window sizes of up to $2^{32}$ - 1 bytes. The window must not be increased above $2^{32}$ - 1 bytes.

A party receiving an APF_ChannelWindowAdjust message on a channel that has been closed, MUST ignore the message.

| Offset (bytes) | Length (bytes) | Description |
|---|---|---|
| 0 | 1 | APF_CHANNEL_WINDOW_ADJUST |
| 1 | 4 | Recipient channel |
| 5 | 4 | Bytes to add |

*Recipient channel* The channel number of the recipient.

*Bytes to add* The number of bytes to add to the window size.

# 4 Main Flows

The following flows and diagrams illustrate the sequences used to for various phases of the APF protocol.

## 4.1 Session authentication and service request state diagrams

The diagrams show the authentication and service request flow between two parties.

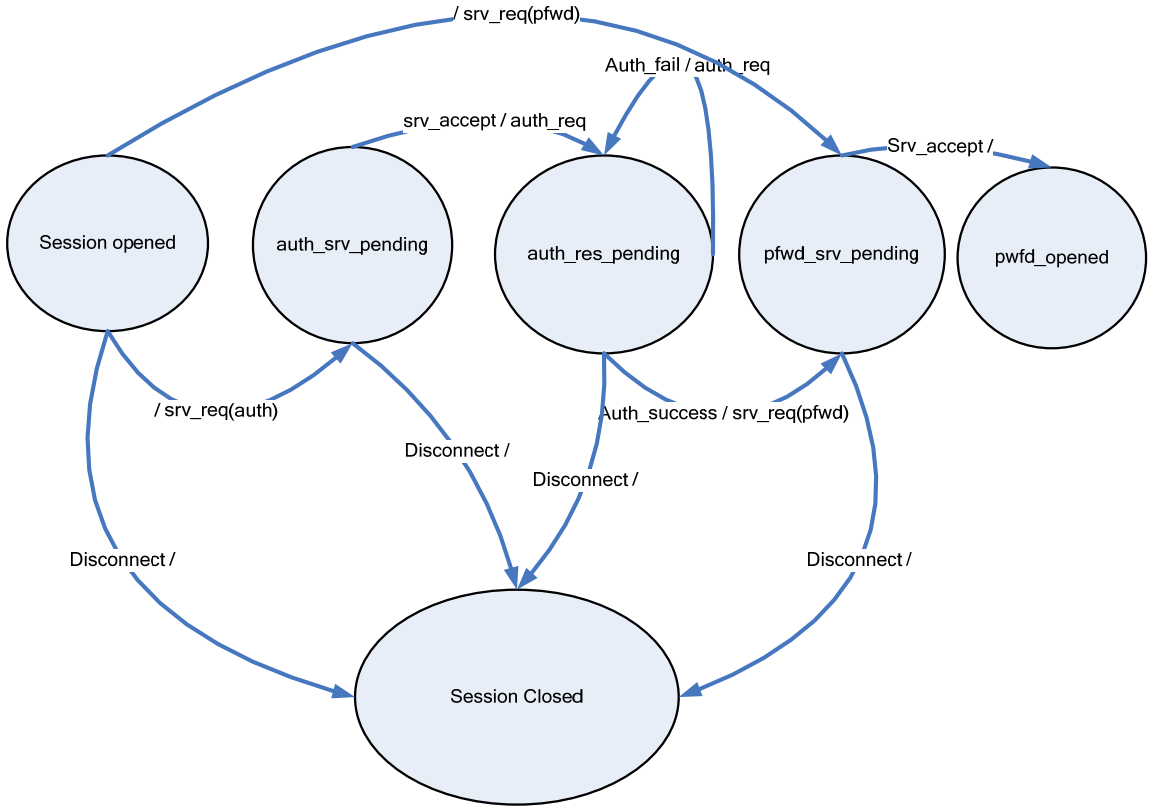


**Figure 1 Client side state diagram**

The diagram illustrates the states a client may pass through to start port forwarding. The client must send an APF_SERVICE_REQUEST prior to sending service–related commands. If authentication is not required, client MAY request the "pfwd" service directly, or use the "none" authentication method.

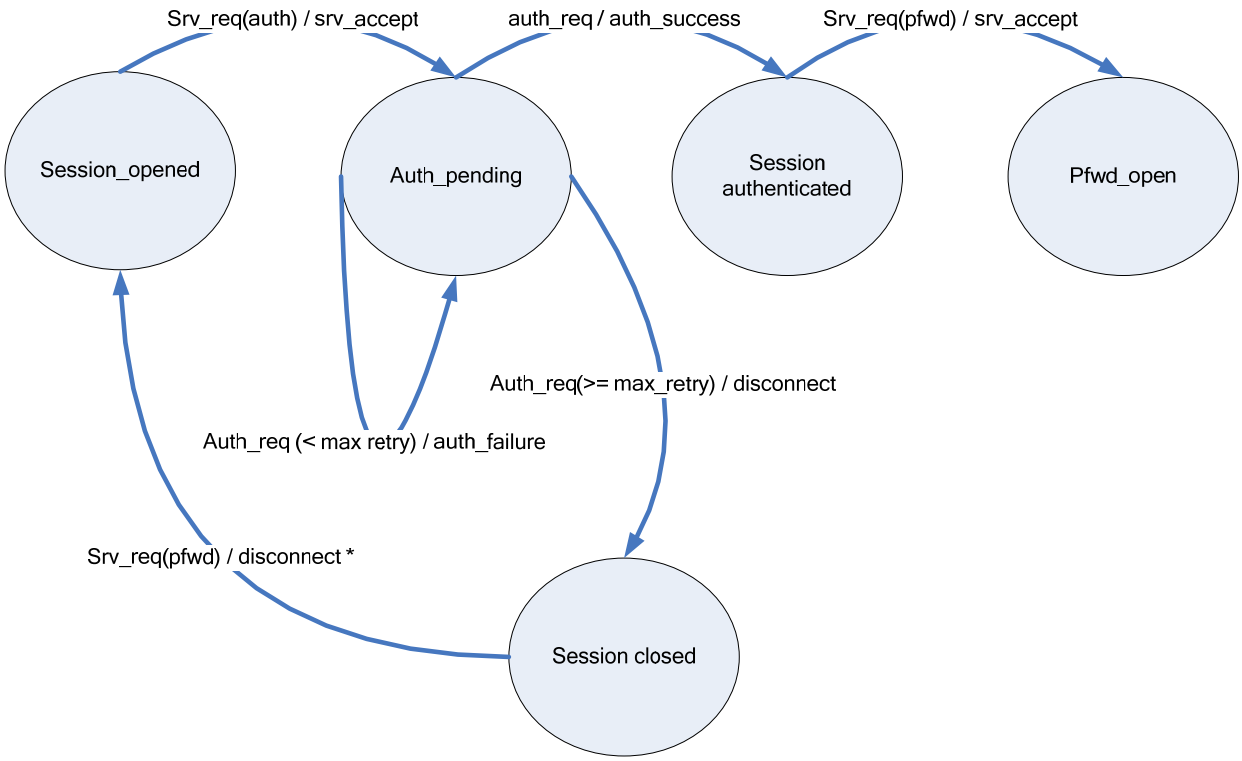


**Figure 2 Server side state diagram – authentication required**

The server side establishment of port forwarding is essentially the same as for the client side.

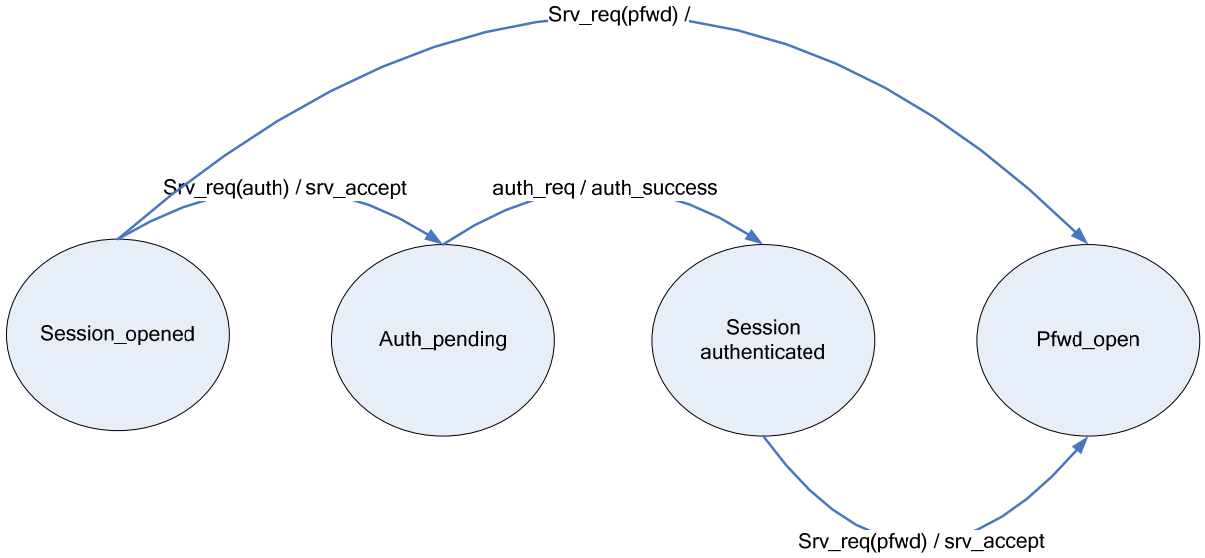


**Figure 3 Server side state diagram – authentication not required**

When authentication is not required, the server can pass directly to the port forwarding open state.

## 4.2 Port forwarding flows

### 4.2.1 Open and Close a Forwarded Channel

A client or server can open a forwarded channel if its peer specifically requested TCP Port-Forwarding in advance. If it had not requested port forwarding, the receiving peer should reject the new channel. When a Port-Forwarding request is canceled, no more channels can be opened. However, existing channels are not automatically closed.

The protocol does not define any limitation to the number of channels that can be opened for a single Port-Forward request. The limitation is specific per implementation.

The following diagrams show the possible states of a client or server regarding a forwarded channel. The initial state is “Disabled” and the progress to the other states depends on messages sent and received by the two peers. When a reply message is needed, the black arrow represents a success status and the red arrow represents a failure. Solid lines represent messages sent by peer A. Dotted lines represent messages sent by peer B. The states are identical in the sending/receiving sides once the message is received.

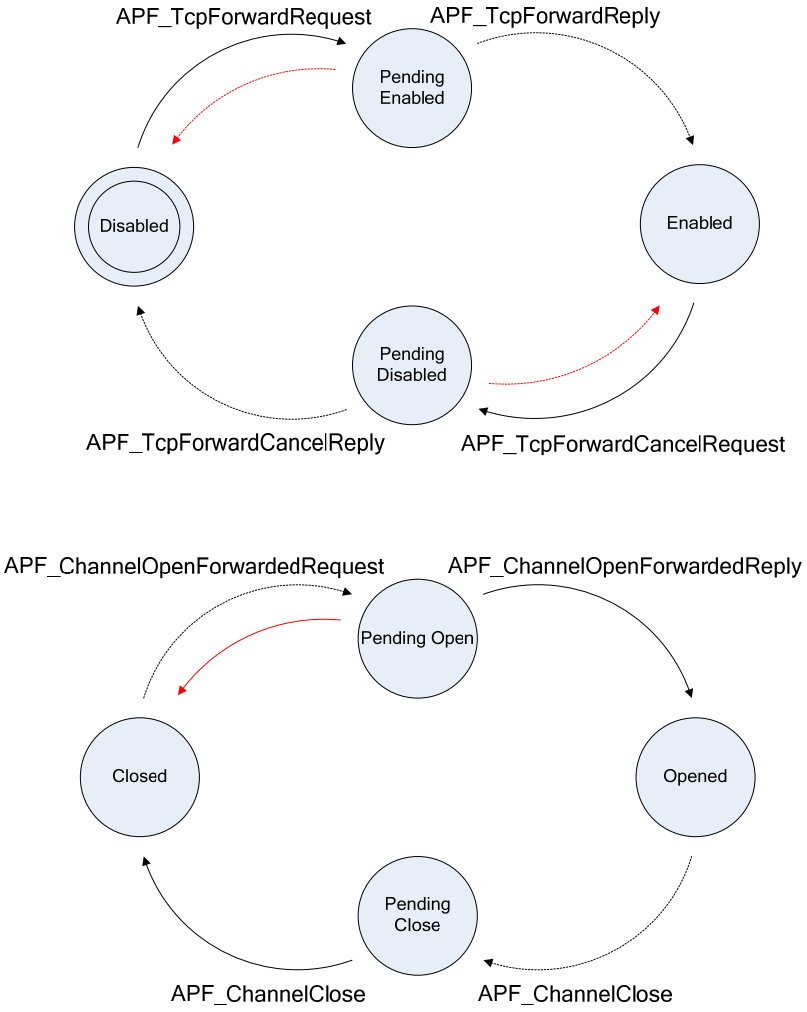


**Figure 4 State diagram for port-forwarded channel**

Detailed flow description:

1. A sends “APF_TcpForwardRequest” message to B.
2. If B cannot accept the request, it will send an “APF_TcpForwardReply” message with fail status to A. Both nodes return to the “Disabled” state.

   If B can accept the request, it will send an “APF_TcpForwardReply” message with success status to A. The state is now “Enabled”.
3. B can send “APF_ChannelOpenForwardedRequest” messages to A only when it is in the “Enabled” state,.

4. If A cannot accept the request, it will send an “APF_ChannelOpenForwardedReply” message with fail status to A. The state is back in “Closed’.
5. If A can accept the request, it will send an “APF_ChannelOpenForwardedReply” message with success status to A. The state is now “Opened”.
6. Both sides can send data to each other once they are in the “Opened” state.
7. Either side can choose to close the channel by sending the message “APF_ChannelClose”. The receiving side must send the same message back for the state to change to “Closed”. Until then, the state is “Pending Close”, which means that the initiator must not send data in the channel but will accept data from the other side.
8. To cancel a port-forwarding request, A will send the message “APF_TcpForwardCancelRequest” to B.
9. If B can cancel the request it will send “APF_TcpForwardCancelReply” to A with a success status. Otherwise, it will send the same message with a fail status.

### 4.2.2 Open and Close a Direct Channel

A direct channel should not be requested in advance. When one of the peers wants to open a direct channel it just sends the “APF_ChannelOpenDirectRequest” message. The channel will open if the other side is willing to accept the new request. This mode is similar to the ‘forwarded channel’ mode once it reaches the “Enabled” state.

The following diagram shows the possible states of a client or server in the direct channel case. The initial state is “Closed” and progress to the other states depends on the messages sent and received by the two peers. When a reply message is needed, the black arrow represents a success status and the red arrow represents a failure.

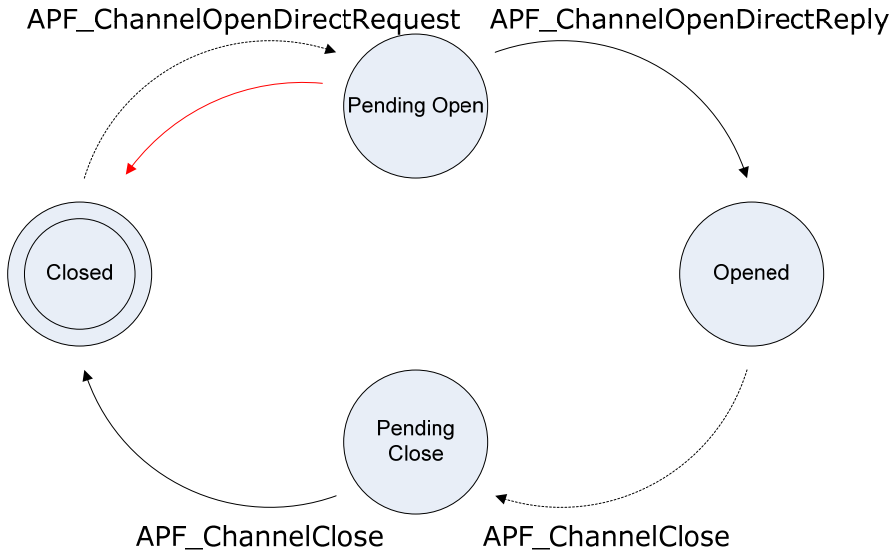


**Figure 5 State diagram for direct channel**

In the diagram above, continues lines represent messages sent by peer A. Dotted lines represent messages sent by peer B.

Detailed flow description:

1. A sends “APF_ChannelOpenDirectRequest” message to B.

2. If B cannot accept the request, it will send an “APF_ChannelOpenDirectReply” message with fail status to A. The state returns to “Closed”.

   If B can accept the request, it will send an “APF_ChannelOpenDirectReply” message with success status to A. The state is now “Opened”.

3. Once both sides are in state “Opened” they can send data to each other.

4. Either side can choose to close the channel by sending the message “APF_ChannelClose”. In the diagram below, it is sent by A. The receiving side must send the same message back in order of the state to change to “Closed”. Until then, the state is “Pending Close” which means that the initiator must not send data in the channel but will accept data from the other side.

### 4.2.3 Data Transfer

The data transfer inside a channel must be synchronized between the two sides. Before sending data, a peer must know that the other peer has enough buffers to read the data. This is accomplished by maintaining a window for both sides. “Window size” messages are sent to indicate a change in the amount of free buffers.

The diagram below demonstrates data transfer between two peers (A and B) inside a Directed Channel.

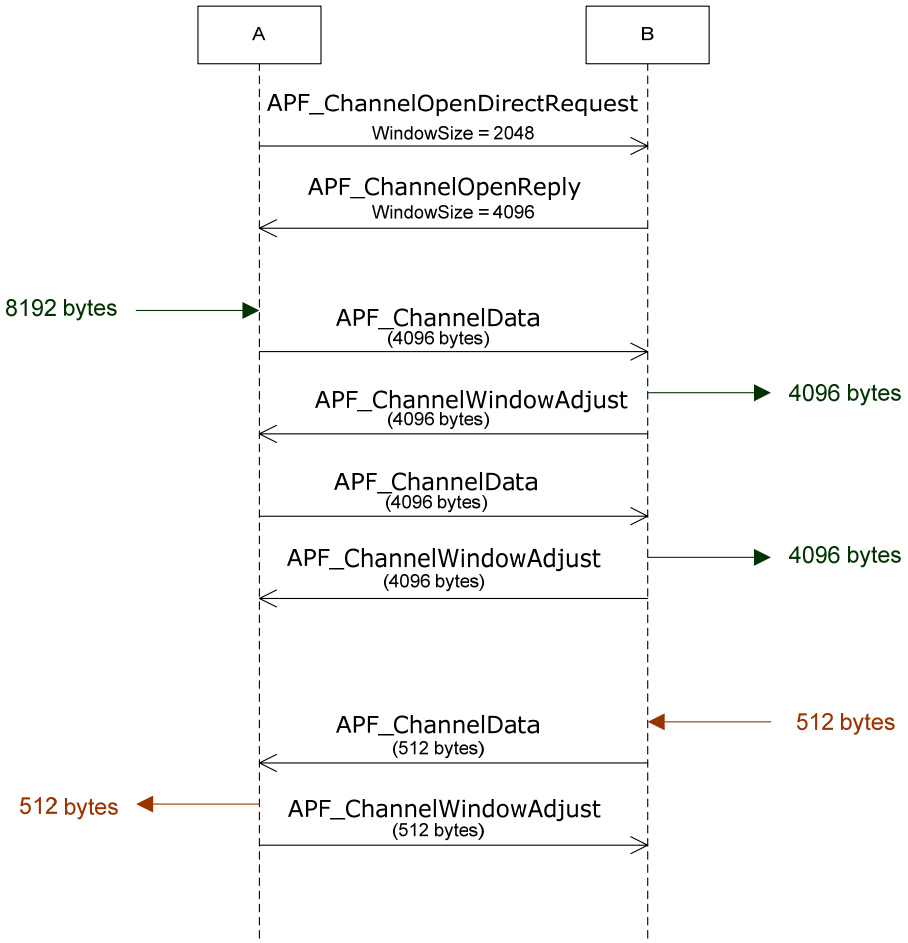

Figure 4: Flow control example

Detailed flow description:

1. A starts a direct channel with B by sending the message “APF_ChannelOpenDirectRequest”. It informs B that the initial window size is 2048 bytes.
2. B accepts the request and sends back “APF_ChannelOpenReply” message. It indicates A about its own window size – 4096 bytes.
3. A receives 8KB of data to push to B. Since B’s window size is only 4KB, it sends an “APF_ChannelData” message with 4KB.
4. B receives the data, pushes it to the relevant client and sends back “APF_ChannelWindowAdjust” message with the change in the window size. The change is 4KB.
5. A can now send another 4KB. B will handle it in the same way.
6. When B receives 512B of data to send to A, it can send it in a single “APF_ChannelData” message since A’s window size is large enough.
7. A processes the data and send back “APF_ChannelWindowAdjust” with a delta, e.g. 512 bytes.

# 5 Intel AMT Use-Cases

The Intel AMT Port Forwarding Protocol as implemented in Intel AMT Releases 4.0 and 5.0, is used in several scenarios. In some scenarios the protocol messages travel over the MEI tunnel (Local Port-Forwarding) and in other scenarios they are travel over a TLS tunnel (Remote Port-Forwarding).

In all scenarios, Intel AMT will never accept a direct channel-open request. Intel AMT only accepts channels which were previously requested to be forwarded. However, as a client, Intel AMT will sometimes request to open a direct channel.

## 5.1 Local Port-Forwarding

Local Port-Forwarding refers to cases where protocol messages are travel over the MEI tunnel. On the host side, the LMS module implements the protocol.

### 5.1.1 Host Agents

In this use-case, a Host Agent needs to connect to an Intel AMT service. This is possible only after it is specifically requested by Intel AMT.

Intel AMT will always allow local port-forwarding for host-agents. Once the LMS MEI client connects to Intel AMT, Intel AMT will send a TCP-Forward request to LMS. The message will contain the following parameters:

- Address to bind = <AMT hostname >
- Port to bind = 16992 or 16993 (Depends on the TLS mode inside Intel AMT.)

Once some host-agent tries to connect to Intel AMT (using LMS), LMS will send a channel-open request (forwarded) to Intel AMT with the following parameters:

- Connected address = “127.0.0.1”
- Connected port = 16992 or 16993.

If the Intel AMT hostname is changed, it will cancel the port forwarding request and open a new one with the new hostname. The cancel message will contain the following parameters:

- Address to bind = <Old AMT hostname>.
- Port to bind = 16992 or 16993 (Depends on the TLS mode inside Intel AMT.)

The LMS will remove the old hostname routing from the ‘hosts’ table. Once a new request is received, it will add the new hostname.

### 5.1.2 Host VPN

In this use-case, Intel AMT is accessible to a remote console over a host VPN connection. All traffic to Intel AMT goes through the LMS. Prior to any connection, Host-VPN must be enabled in Intel AMT. Also, Intel AMT must verify that it cannot access the corporate network directly. The LMS must verify that it can access the corporate network. The method for detecting the connection mode is not in the scope of this protocol.

Once Intel AMT detects it is connected outside the enterprise network, it will send a TCP-Forward request to LMS. The message will contain the following parameters:

- Address to bind = “0.0.0.0”.
- Port to bind = 16992 or 16993 (Depends on the TLS mode inside Intel AMT.)

Once a remote-console attempts to connect to Intel AMT over the host VPN, LMS will send a channel open request (forwarded) to Intel AMT with the following parameters:

- Connected address = Host IP address inside the enterprise network.
- Connected port = 16992 or 16993.

Once Intel AMT detects that the connection to the enterprise network over the VPN is not needed, it will cancel the port forwarding request. The message will contain the following parameters.

- Address to bind = “0.0.0.0”.
- Port to bind = 16992 or 16993 (Depends on the TLS mode inside Intel AMT.)

Note that the cancel request does not cancel the port-forwarding request for the local IP address.

If the Intel AMT DNS suffix list is changed, it will cancel the port forwarding request and open a new one. This means that LMS must query for the DNS suffix list each time a TCP-Forwarded request is received from Intel AMT for address “0.0.0.0”.

## 5.2 Remote Port-Forwarding

Remote port-forwarding occurs in the case where the protocol messages travel over a TLS tunnel. Intel AMT creates the tunnel to a predefined server. A management presence server implements the server side of the protocol.

Once Intel AMT establishes a TLS tunnel, it will authenticate itself if the TLS tunnel is configured for Server authentication. Then Intel AMT will send two TCP-Forward requests to the remote server. The requests will make the server bind to two ports (one for HTTP/HTTPS and one for SOL/IDER). The messages will contain the following parameters.

- Address to bind = <AMT hostname>.
- Port to bind = 16992 or 16993. (Depends on the TLS mode inside Intel AMT.)
- Port to bind = 16994 or 16995. (Depends on the TLS mode inside Intel AMT.)

Before closing the TLS tunnel, Intel AMT will cancel the port forwarding requests using the same parameters.

If the Intel AMT hostname is changed, it will cancel the port forwarding requests and open new ones with the new hostname. The cancel messages will contain the following parameters.

- Address to bind = <Old AMT hostname>.
- Port to bind = 16992 or 16993. (Depends on the TLS mode inside Intel AMT.)
- Port to bind = 16994 or 16995. (Depends on the TLS mode inside Intel AMT.)

The remote server will remove the old hostname routing from its routing table and add the new one instead.

Note that the Intel implementation of the MPS does not support an APF_TcpForwardCancelRequest with Address to Bind string value of "0.0.0.0".

If an Intel AMT platform sends a username/password combination that fails authentication, the MPS sends an APF_REQUEST_FAILURE response followed by an APF_DISCONNECT. It does not wait for an additional authentication attempt.