ZDP Protocol Definition

Frank Kastenholz, September 2, 2025

Contents

List of Figures

List of Tables

1. Introduction

This document defines the ZPR-2020 Data Protocols (ZDP). These protocols 1) manage the interactions of Adapters and Nodes with their immediate neighbors and 2) transport Endpoint Packets from Ingress Adapter to Egress Adapter.

ZPR-2020 operates over substrate networks. A separate document will describe how ZDP operates over various substrate networks.

This note has one major function – to define the ZPR Data Protocol which transports Endpoint Packets, from the source Endpoint to the Destination Endpoint, over one or more Substrate Networks.

This note does not discuss the protocols between Endpoints (other than making the assumption that Endpoints use either IPv4 or IPv6 to communicate) between an Endpoint and its Adapter, within ZPR Network Services, or between a Network Service and a ZPR Forwarder or Dock.

1.1. Conventions

All numeric values are in network byte order (that is, big-endian).

The terms MUST/MUST NOT, SHOULD/SHOULD NOT, REQUIRED, SHALL, SHALL NOT, RECOMMENDED, OPTIONAL, and MAY have the meanings typically used in IETF RFCs (see RFC 2119[1]).

The phrase “silently discarded” (or equivalent) is often used in the protocol descriptions. Paraphrasing the Host Requirements RFC, RFC1122[2]: This means that the packet will be discarded without further processing and that an error indication packet will not be sent as a result. However, for diagnosis of problems the Node SHOULD provide the capability of logging the error, including the contents of the silently discarded packet and SHOULD record the event in a statistics counter. Unless otherwise specified, higher level policies control further actions that may be taken as a result of such events.

“IP” means “IP of either version” -- “IPv4 and IPv6” or “IPv4 or IPv6” (the context should make it clear which is meant). If a specific version of IP is intended, then it will be explicitly stated: “IPv4” or “IPv6”.

1.2. Terminology

ZPR makes use of specific terms in specific ways. ZPR-specific terms (or use of general terms in ZPR-specific ways) is denoted by capitalizing the term. The (uncapitalized) term “link” could be any form of physical connection between two boxes, or even a sausage. “Link” however is the entity that connects one ZPR Forwarder to another and is used by the Forwarders to forward traffic to a destination.

Most terms are defined in ZPR RFC 4[3]. Other terms and abbreviations are:

[1] Bradner (1997)
[2] Braden (1989)
[3] Douglas and Kastenholz (2020)

Upstream and Downstream Refers to the relative positions of Docks and Forwarders with respect to a specific flow of packets that are being transported through the ZPR network. The Upstream Dock or Forwarder sends packets to the Downstream Dock or Forwarder. Endpoint (that is, data) packets are always sent downstream. Error notifications (*e.g.* can't forward a data packet because a Link has gone down) would typically be sent Upstream.

Ingress and Egress Refer to the Docks and Adapters where Endpoint Packets enter the ZPR Network and leave the ZPR network, respectively.

Source and Destination The ultimate source and destination of an Endpoint Packet. Typically, these packets are IP packets whose source- and destination-IP addresses are the Endpoint Addresses of the source- and destination Endpoint.

D2D Dock to Dock (typically used for Dock-to-Dock security) (Obsolete)

A2A Adapter-to-Adapter, used to indicate Adapter-to-Adapter Security and SA.

SAID Security Association ID

MICV (also MAC) Message Integrity Check Value (or. Message Authentication Checksum). The two are used interchangeably in this note.

PEP Policy Enforcement Procedure

AA Endpoint -Address

D2DSA Dock to Dock Security Association (Obsolete)

A2ASA Adapter to Adapter Security Association

Policy A Policy can be

1) A set of rules that are interpreted by the Admin and Visa services in order to determine whether a Visa should be granted or not and, if it is granted, the parameters of the Visa. Policies may be interpreted multiple times, possibly leading to the granting of multiple Visas or

2) A set of rules that controls the operations of other elements of the ZPR Network.

Visa A Visa is a specific permission for a related set of communications, or Streams, to occur. The Visa and the individual streams define how those communications occur (such as the path through the ZPR network, rate limitations, and so on). They also define conditions under which the Nodes should terminate a Visa or Stream. An example of the Streams that a Visa may permit are the forward and backward traffic flows of a specific TCP connection, along with any supporting ICMP traffic.

Visas have globally unique names.

Stream A Stream is a single unidirectional flow of traffic from a specified source Endpoint to a specified destination Endpoint. The Stream includes classification rules (the Traffic Classifier) to exactly identify traffic to flow in the stream.

Streams have names that are unique within the context of the Visa that provisions the Stream.

Flow A Flow is the actual sequence of bytes going from the source Endpoint to the destination Endpoint. It is carried in ZDP Transit packets. This is sometimes referred to as a Traffic Flow.

1.3. Version

This document describes version 1 of ZDP.

2. Protocol Stack

The following diagram shows the abstract ZPR-2020 protocol stack, showing the relationships between the various elements.

```
+-------------------+
|   Endpoint        |
|    Application    |
+-------------------+
|   Endpoint        |
|    Application    |
|     Protocol      |
+-------------------+
|   TCP, IP, Etc.   |
+-------------------+
|   Endpoint Packet |
|    Messages       |
+-------------------+----------------+
|                   |                |
| ZPR Data Protocol | ZPR Management |
| (ZDP)             | Messages       |
+-------------------+----------------+
    +--------+-+--------+ - - - - - - -
    |        | |        |  Adaptation
    |        | |        |  Layer
    +--------+ +--------+ - - - - - - -
    |        | |        |
    |        | |        | Substrates
    |        | |        |
    +--------+ +--------+ - - - - - - -
```

Figure 2.1: Abstract ZPR-2020 Protocol Stack

Where:

Endpoint Application Is the application software that is the source or sink of Endpoint data. When combined with other identity attributes it is a ZPR Endpoint. Examples might be web browsers and servers, email clients and servers, and so on. Endpoint Applications may have a client/server or peer-to-peer relationship; ZPR does not distinguish.

Endpoint Application Protocol Is the protocol that the Endpoint Application uses to communicate over its network connection. Examples include HTTP or SMTP.

TCP, IP, Etc., The lower layer protocols over which the Endpoint Application Protocol operates.

The Endpoint Application Protocol and "TCP, IP, etc." together generate and receive Endpoint Packets. Endpoint Packets are generated by the Source Endpoint Application and are transported by ZPR Transit Packet Messages to the Destination Endpoint Application.

ZPR Management Messages Are ZPR messages that are used to control the Link or Docking Session over which ZDP operates.

ZPR Data Protocol (ZDP) This protocol operates between linked ZPR Nodes, as well as between Adapters and Docks. It provides security and demultiplexing services (§ 4.2).

Adaptation Layer This layer provides whatever functions are needed for ZDP to operate over a given substrate technology. For example, if the substrate is a point-to-point fiber-optic link then the adaptation layer may be something like IETF Standard PPP[1] in order to provide needed demultiplexing and link state synchronization. In many cases the Adaptation Layer will be minimal, or even NULL. We include it, however, for completeness.

Substrate The Substrate is whatever it is that ZDP runs over. For ZPR-2020 this is an IP network of some form or a packet-oriented data link of some type.

[1] Simpson (1994)

3. Procedures

This section describes various ZDP procedures. It starts with descriptions of typical ZPR Nodes and Endpoint/Adapters. These descriptions provide the context in which the procedure descriptions are given. The descriptions then follow in roughly the order they might occur in a Node or Endpoint.

3.1. Notional Node Model

A Notional Node typically contains a Forwarder, zero or more externally-facing (that is, connecting to external Adapters) Docks, at least one internally-facing Dock used to serve the ZPR trusted services (Admin, Visa, and so on) which manage the Node, and one or more Substrate network interfaces over which Links and Docking Sessions are to be instantiated. In addition, the Node contains elements of the services which comprise the ZPR Network, such as the Admin Service, Visa Service, and Authentication Service. These elements are ZPR Endpoints and communicate with their counterparts on other Nodes and back-end servers (which are ZPR nodes in their own right) via their own protocols, operating over ZDP.

```
+------------------------------------------------------------------+
|                                                                  |
|  +----------+  +----------+  +----------+                        |
|  | Visa     |  | Admin    |  | Auth     |                        |
|  |  Service |  |  Service |  |  Service |  +-------------------+ |
|  +----------+  +----------+  +----------+  | External          | |
|  | Adapter  |  | Adapter  |  | Adapter  |  |  Facing           | |
|  +----------+  +----------+  +----------+  |   Dock            | |
|Internal :    Docking  :  Sessions  :       |                   | |
|  +----------+---------------------------+  |                   | |
|  |      Internal Dock                   |  |                   | |
|  +--------------------------------------+  +-------------------+ |
|Virtual :                                 Virtual :     :         |
|   Link :                                    Link :     :         |
|        :  +------------------------------------+ :     :         |
|        :  |          Forwarder                 | :     : Dock    |
|        :  +------+------+------+------+--------+ :     : Sess.   |
|        :  | Link | Link | Link | Link | Link |   :     :         |
|        :  +------+------+------+------+------+   :     :         |
|        :     :       :      :      :       :     :     :         |
|        .......       :      :      :       .......     :         |
|                      :      :      :                   :         |
|      +------------------------------------------------------+    |
|      |           Substrate Network Interfaces               |    |
|      +------------------------------------------------------+    |
+------------------------------------------------------------------+
```

Figure 3.1: Notional Node Structure

:'s and .'s represent the ZDP data flows. The Link between the External Facing Dock and the Forwarder is shown as a “Virtual Link”. In this description, the link is not a physical network connection but rather some kind of software, function-call or message-passing API. It may also be implemented on a separate platform, with a physical network link between the Dock and Forwarder.

Where:

Visa, etc., ZPR Trusted Service These are the components of the ZPR Trusted services that reside on the Node and provide the relevant services to the Node. They are Endpoints.

Adapter Internal Adapter to connect the internal ZPR Trusted Services to the ZPR Network.

Internal Docking Sessions Are Docking Sessions between the Internal Dock and Internal Adapters for the ZPR Trusted Services.

Internal Dock Dock functionality that faces the internal Trusted Services.

External Facing Dock This Dock faces external Adapters, connecting them to the ZPR Network.

Forwarder Is a forwarder, switching ZDP Transit Packets among the different Links (regardless of whether the Link is internal or not).

Link The Link and Adaptation layer.

Virtual Link A Link that is internal to the Node, connecting the Docks to the Forwarder. These Links are not physical networks. Typically, they are message passing or function call software APIs.

Substrate Network Interface This is the interface to the substrate network. Separate substrate networks have separate interfaces.

Docks and Forwarders are presumed to contain the following tables:

Dock Forwarding Table (DFT) The Dock contains several DFTs. Each Docking Session and Dock/Forwarder Link has its own DFT. The lookup key is the Stream ID of a packet received on the Link or Docking Session. The result is a Policy Enforcement Procedure (PEP), that specifies how the packet is to be handled -- which Docking Session or Link that the packet is to be transmitted on and the Stream ID to put into the transmitted packet.

Forwarding Table (FT) The Forwarding Table is similar to the FIB (Forwarding Information Base) in classic IP routers. It is used by ZPR Forwarders to determine how a ZDP Transit Packet should be handled.

Each ZPR Configuration that is operational in a ZPR Network MAY have its own Forwarding Table. Conversely, multiple operational configurations may all use a shared FT. Regardless, each Configuration MUST have a FT, whether shared or private.

The FT takes, as input, the Stream ID from a received ZDP Transit Packet and an identifier of the Link on which the packet was received. It produces a Policy Enforcement Procedure (PEP) describing how the packet is to be handled. PEP typically contain the following:

- Link out of which the packet should be forwarded. Note that this Link may be an internal Link to a Dock on the Node, including the Internal Dock (and thereby the ZPR Trusted Services).
- Stream ID to insert in the packet when it is transmitted,

- Traffic constraints (such as maximum number of bytes or packets that may be transmitted in a given time period, maximum transmit rates, and so on),
- QoS considerations when transmitting the packet, such as pacing or scheduling the traffic, Substrate network QoS marking (such as DSCP on IP Substrates), transmit queue assignment, and so on.

In addition, the PEP may also indicate other dispositions of the packet such as

- Silently discarding the packet.
- Discarding the packet and sending a specified Destination Unreachable (§ 4.3.13) Management Message.

It might seem reasonable to specify that if a lookup in the FT fails there is a default PEP that specifies that a Destination Unreachable message of some form always be sent upstream. We do not do this since sending a Destination Unreachable might reveal sensitive information to an attacker. Thus, we allow ZPR Policy to determine the action taken when a lookup fails (of course, the policy may direct that a Destination Unreachable be sent).

There are internal APIs between the various service Endpoints, the Docks, Links, and the Forwarder. These APIs are used to pass management, control, and reporting information among the various entities. These APIs are implementation specific and not specified in this document. For example, if a Link detects a failure, it informs the Admin Service of this via an internal API.

> **Note Well**: The Node described above is notional and serves as context in which ZPR operations may be specified. It is not meant to be prescriptive nor an actual implementation design. Implementors may design their Nodes in any way, so long as the functions and processes described below perform as specified.

Adapters contain two lookup tables, the Endpoint Lookup Table and the Dock Lookup Table,

Endpoint Lookup Table (ELT) Packets received from the Endpoint are looked up in the Endpoint Lookup Table. This table performs a typical IP packet classification. That is, the packet is looked up using its IP Addresses and, possibly, protocol, and port numbers of the packet and produces a PEP that specifies how the packet is to be compressed and what Stream ID to use when sending the packet to the Dock.

If a packet is not found in the ELT then it indicates that a new flow is to be created; the “Adapter to Dock Tether Establishment and Endpoint Packet Transmission” procedure, § 3.7, is performed.

Dock Lookup Table (DLT) The second table, the Dock Lookup Table, reverses the operation; the Stream ID from received Transit Packets is looked up in the table, producing a PEP specifying the Endpoint IP addresses, protocol, and port numbers as well as decompression instructions.

3.2. General Setup Operations

This section provides the high-level sequence of events for bringing Nodes and Endpoints up and making them operational. This sequence is not mandatory. Implementations may differ from this sequence. The sequence of operations is:

1) Forwarders establish links between each other as directed by Connection Policies.
2) Docks prepare to receive in-bound Docking Sessions from Adapters, as directed by Policies.
3) An Adapter establishes a Docking Session with a Dock. As a part of this, the Adapter registers the Substrate Addresses (typically IP addresses) with the Dock via the Register Endpoint Address message (§ 4.3.8). The Dock passes the Endpoint Address(es) as well as the Dock's ZPR Address to the Topology Service.

   The stimulus for establishing Docking Sessions varies. Some stimuli are:

   - Attempting to send a ZDP message to a particular Dock when there is no existing Docking Session, or
   - Done automatically, driven by Policy, regardless of whether there is data to send or not. This may be done by servers.
4) A source Endpoint sends the first Endpoint Packet to a destination Endpoint. The destination address in this packet is the Endpoint Address of the destination Endpoint (the source address is the Endpoint Address of the source Endpoint). This packet is received by the ingress Adapter.
5) The Ingress Adapter requests a Stream ID for the Endpoint IP Addresses, IP protocol and UDP or TCP ports (if the packet is UDP or TCP) tuple from the Dock. The Bind Endpoint Address message is used for this (§ 4.3.9).
6) The Dock:
   a. Checks its tables to see if the packet is covered by an existing Visa. If so, the Dock responds to the Bind Endpoint Address request with the Stream ID allocated for that Visa.
   b. If there is no satisfactory Visa, the Dock requests a Visa from the Visa Service. The Visa Service performs whatever operations are necessary to evaluate the request and grant a Visa. In particular, the Visa Service may do a ZPR Authentication of the Endpoint using the Authentication Messages. See also § 3.26.
   c. The Visa Service issues a Visa and installs it on the Ingress Dock, intermediate Forwarders and Egress Dock.

      The Visa Service uses the Topology Service to determine the path through the network that the packets should take. It knows the Dock to which the destination is connected because the destination has registered its Endpoint Address when it connected to the ZPR Net (see step 3).
   d. The Dock responds to the Bind Endpoint Address request with an appropriate Stream ID.
7) The Adapter builds appropriate tables to forward Endpoint Packets of the Flow via the Stream ID it received from the Dock.
8) As the first Endpoint Packet of the Flow traverses the network, each Node requests the Stream ID to use in the packet from the next hop (§ 3.11). The requests identify the desired Stream by the Visa and Stream names. The Downstream Node responds with the Stream ID to use when forwarding packets via the Stream as permitted by the Visa.

A Downstream Node MAY prospectively push the Stream ID for a Visa to the Upstream Node prior to arrival of a Transit Packet. This can reduce latency for the first packet. See § 3.12.

9) The Endpoint packets are forwarded through the network (§ 3.14).
   a. The Ingress Adapter will compress the Endpoint Packets, removing fields from the Endpoint Packet as directed by the Dock in Endpoint Bind Endpoint Address response.
   b. The Ingress Adapter generates the A2A MICV for the Endpoint Packet. The SA parameters (keys, algorithms, and other parameters) are received from the Visa Service and sent to the Adapter in the Bind Endpoint Address response. The keys are encrypted by the Visa Service using a key known only by the Adapter (*e.g.* it may be generated as a result of the Authentication operation or by the Adapter's public key) and sent to the Adapter in the Bind Endpoint Address response.
10) When the packet reaches the Egress Adapter the Adapter checks the MICV, decompresses the Endpoint Packet and then sends the packet to the Endpoint. NOTE that the IP addresses, protocol, and transport ports that are compressed out of the packet by the ingress Adapter, are reinserted based on the values the egress Adapter receives in the Visa.

3.3. Link and Docking Session Finite State Machines

ZPR Links and Docking Sessions operate using simple finite state machines (FSMs). The FSMs for each are only slightly different and the description of them is as common as possible (in order to avoid repeatedly defining the same events, states, or transitions with the possibility that the definitions, which are intended to be the same, in fact diverge).

The following three diagrams show the primary state transitions for the FSMs for managing the state of a Forwarder to Forwarder Link, the Dock side of a Docking Session and the Adapter side of a Docking session, respectively. Tbls. 3.1, 3.2, 3.3 show the full FSM specification.

3.3.1. States

Initial The initial state of the Link or Docking Session. It can do nothing in this state except receive a configuration.

This is the "power on" condition of the Link and Docking Session.

Any packets received while in this state MUST be silently discarded. They are not considered to be errors or policy violations.

Inactive The Link or Docking Session has been configured but has not started. The Link or Docking Session is started only under certain conditions, as defined by ZPR Policies. When those conditions occur, a START event is issued to the relevant Link or Docking Session.

Any packets received while in this state MUST be silently discarded. They are not considered to be errors or Policy violations.

When this state is reached:

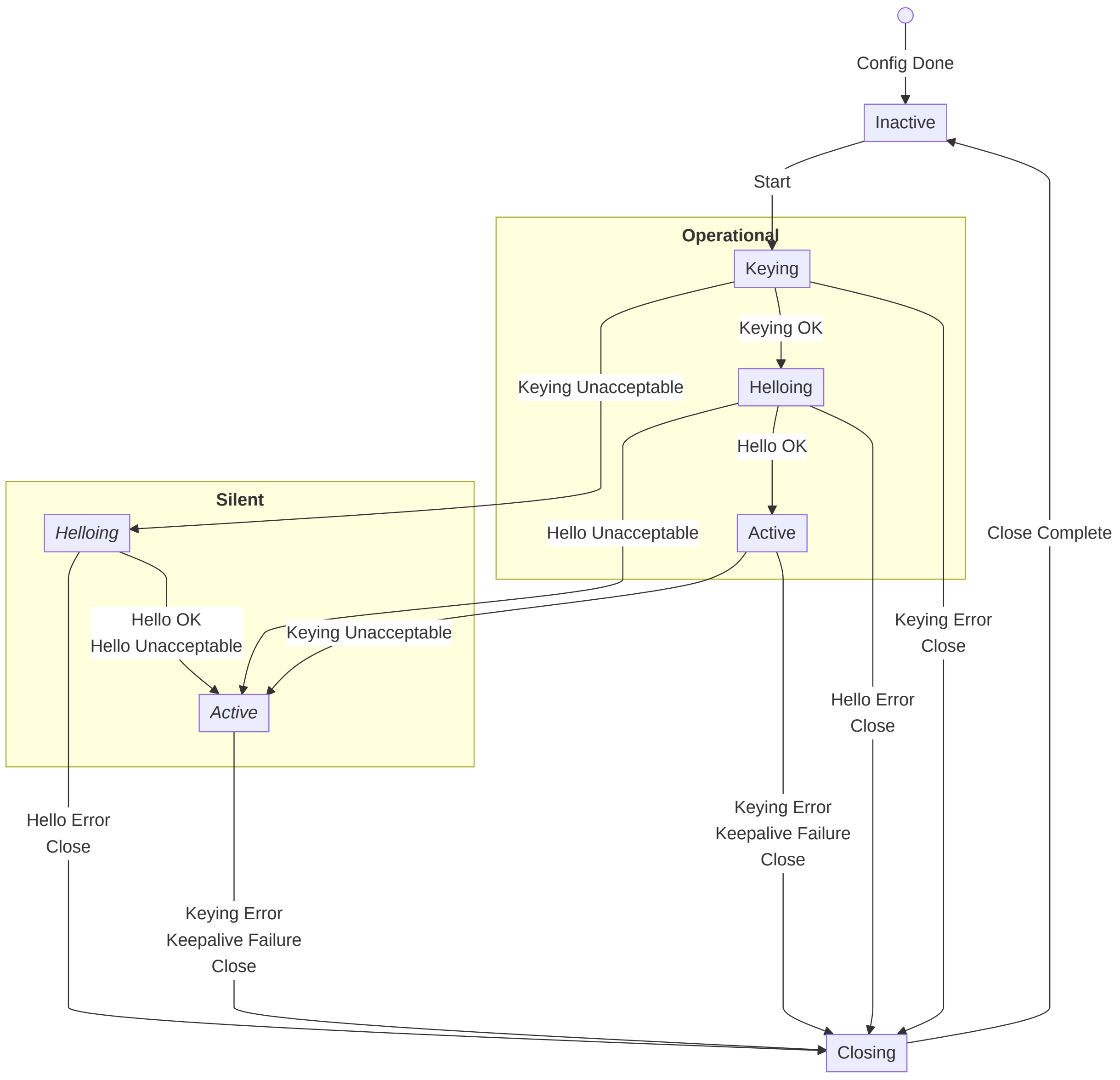


Figure 3.2: Link FSM

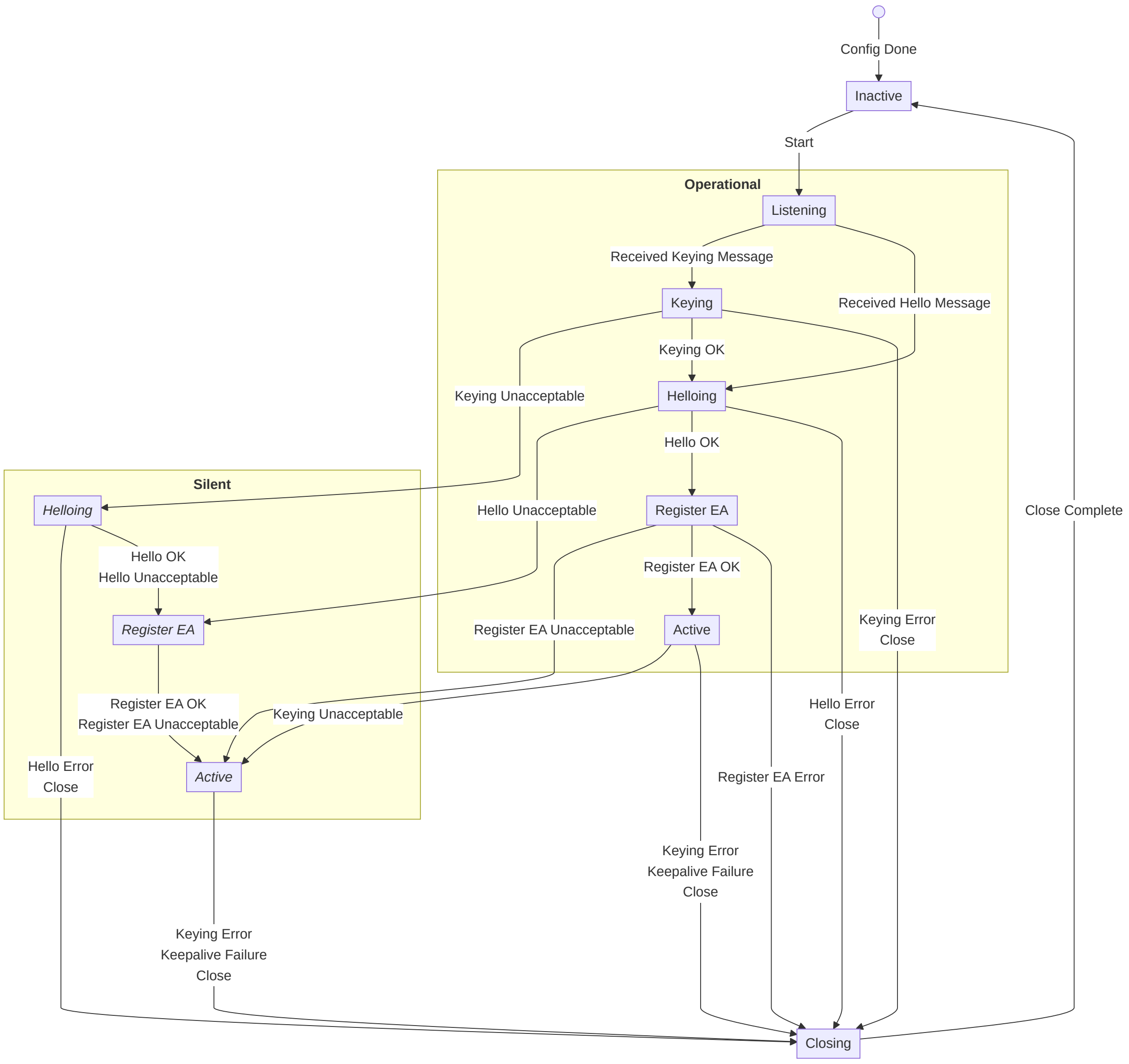


Figure 3.3: Docking Session FSM (Dock Side)

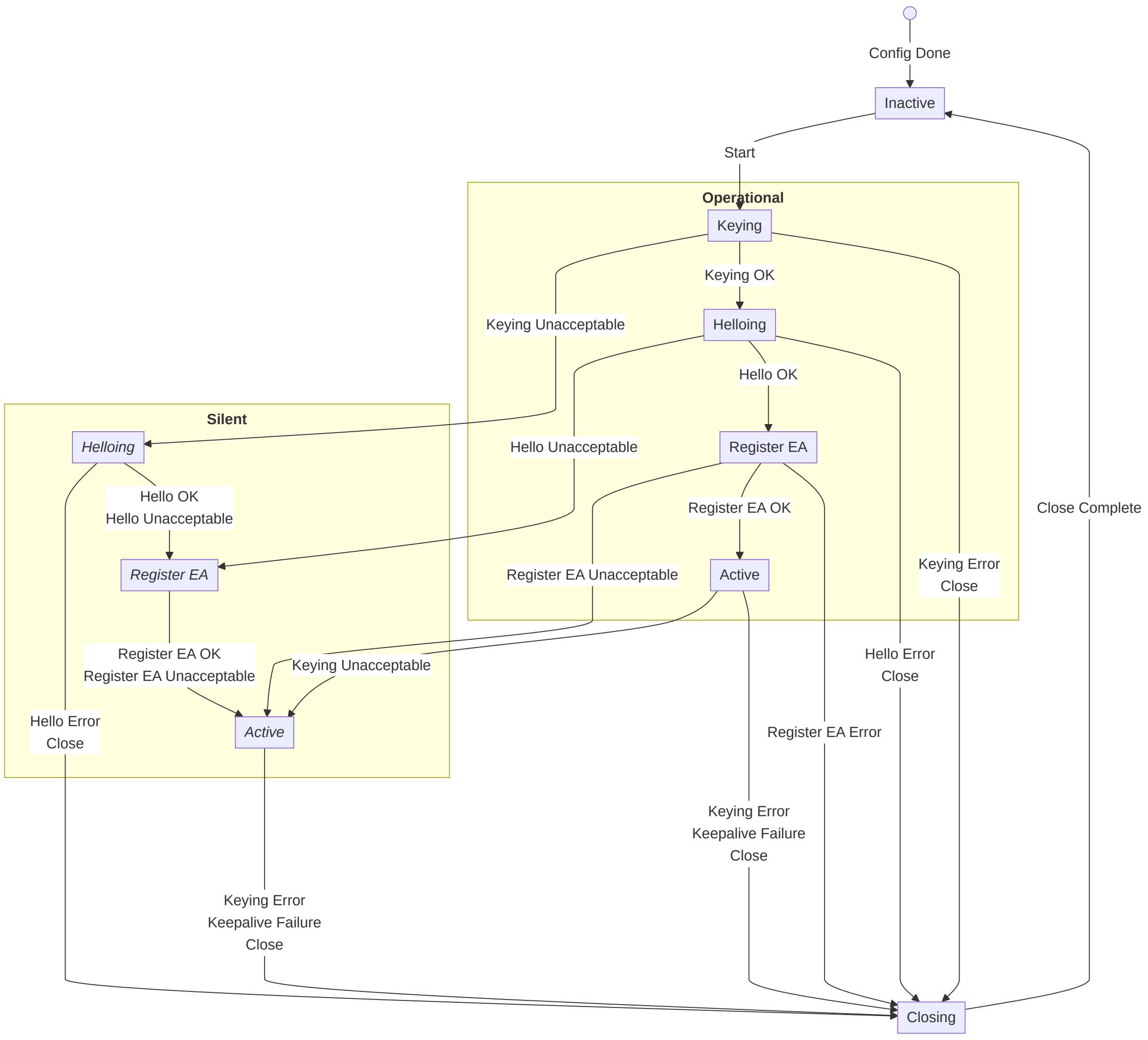


Figure 3.4: Docking Session FSM (Adapter Side)

1) If keys are predistributed then the configuration MUST have created the Security Associations (SAs) to be used,
2) If dynamic keying is done, then SA 0 (the NULL SA, § 4.2.2.1) MUST be created and available to process keying messages. Keying

If predistributed keys are used, then this state is a NO-OP. The Dock, Adapter, or Forwarder notes that an SA is available and automatically issues a KEYING DONE, OK event (This eliminates a special case in the FSM).

If a keying protocol is used, then that protocol executes. The two sides of the Link or Dock Session attempt to establish an initial Security Association (SA) to use for subsequent messages.

The Keying Protocol has its own, internal FSM, that it uses to establish the SA. We assume that this FSM can produce one of three results:

- SA Established – the SA has been created and is available for use. A KEYING_DONE, OK (KDo) event is then generated.
- Error – some (unrecoverable) error has occurred. The error should be reported, state cleaned up and resources recovered, and the Link or Docking Session returned to its previous state. These errors are things like running out of memory, not being able to agree on an algorithm, or network failure. In this case a KEYING_DONE, Error (KDe) event is generated.
- Unacceptable – the operation terminates but the peer is deemed to be unacceptable for some policy-based reason (*e.g.*, a certificate has been revoked or the signer is unknown or unacceptable). In this case a KEYING DONE, Unacceptable (KDu) event is issued.

NOTE WELL: additional keying protocol operations (such as updating keys, managing SAs, etc) may occur, as needed by the keying protocol, but only after the KEYING state has terminated successfully.

Helloing The HELLOING state is used by the peers to exchange information: the version of ZDP that the peer is using and the ID of the policy by which the peer has started the Link or Docking Session.

The peer receiving the hello information evaluates the information. If it is deemed unacceptable for any reason, the peer terminates the Docking Session or Link but “goes silent” with respect to the peer (that is, any Transit Packets sent by the peer are silently discarded).

The Hello operation performs its own, internal, FSM. It can have one of three results:

1) OK. The hello operation completed, all data is acceptable, and the Link or Docking Session can proceed to the next step. In this case, the HELLOING DONE, OK (HDo) event is issued.
2) Error – Some kind of error occurred, and the operation is deemed to have failed. Resources are recovered and the Link or Docking Session returns to the INACTIVE state. In this case, the HELLOING DONE, Error (HDe) event is issued.

3) Unacceptable – The operation completed but the information in the Hello message was unacceptable for some reason AND the disclosure of the fact that the information wasunacceptable could leak sensitive Policy or Configuration information. In this case the HELLOING DONE, Unaceptable (HDu) event is issued.

Closing The Link or Docking Session has started to close. Forwarding has stopped and resources are being recovered. When the close completes, the Link or Docking Session goes to the INACTIVE state.

While in this state:

- Any buffers containing packets received on or scheduled for transmission out of the CLOSING Link or Docking Session are silently discarded,
- Any QOS or other network resource reservations or allocations on the Substrate network or interface over which the Link or Docking Session operates are released,
- Any Endpoint Addresses that have been registered on the Docking Session are withdrawn from the ZPR network (Dock only),
- Any security associations established by a keying protocol are terminated.

Helloing Silent In this state, the node pretends to complete the helloing operations successfully, regardless of whether the hello information received from the peer is acceptable or not. It is entered only if the keying completes unsuccessfully (KDu). The intent is to fool the peer into thinking that "all is well" even when it is not (and thereby not revealing any sensitive policy, etc, information). The HD event is issued when the helloing finishes.

Active All initial configuration and startup have been done. ZDP packets are being forwarded via the link.

Additional Keying and Endpoint Address registration operations may occur. If either results in an error (KDe or RADe) then the Link or Docking Session is terminated as if a CLOSE event had been issued. If the operations terminate unacceptably (that is, some part of the operation would violate policy or otherwise expose sensitive information to a potential attacker) then the KDu or RADu events are issued and the Link or Docking Session transitions to the Active Silent state.

Active Silent In the Active Silent state node appears to the peer to be in ACTIVE state, but in fact, the node discards all ZDP transit traffic received from the peer and discards any transit traffic it would have sent to the peer. Active Silent is only entered when some management operation (*e.g.*, establishing new keys) fails due to a Policy violation and exposing this failure to the peer would expose the Policy to a potential attacker. The intent is that the node then "pretends that everything is normal" so as not to reveal the failure to the peer.

Any management messages received from the peer are responded to in an affirmative manner (that is, as if the operation succeeded), though the operation(s) that would result from the message are not performed.

Traffic received from the ZPR network (or Endpoint) to be transmitted via the Link or Docking Session MUST NOT be so transmitted. It MUST be dropped.

The node MAY elect to conduct a more complex deception by sending an appropriate ICMP Destination Unreachable in response to receiving packets via the Link or Docking Session.

Listening This state applies only to the Dock side of a Docking Session.

In this state, a Dock is waiting for an Endpoint/Adapter to attempt to connect to the ZPR Network. The Endpoint/Adapter initiates the connection by sending either the first keying protocol message (in which case a Receive Keying Message (RKM) event is issued) or the first message of the hello protocol (in which case a Receive Hello Message (RHM) event is issued).

This state is required (and limited to Docks) since Docks passively wait for Adapters to start connecting to the network.

Register EA The REGISTERING EA (Endpoint Address) state is used only by Adapters and Docks with respect to individual Dock Sessions.

When an Adapter has successfully established keys and completed the hello operation, it MUST register at least one Endpoint Address with the Dock. This performed in the Registering Endpoint Address state.

The Register EA state operates its own internal FSM. It can produce one of three results:

1) Endpoint Address registered successfully. A Register EA Done, OK (RADo) event is issued.
2) Error occurred while registering address (such as no memory or network error). In this case the operation terminates and a Register EA Done, Error (RADe) event is issued.
3) Attempt to register an address that violates a Policy or the success/failure of such a registration would reveal sensitive internal state of the network to a potential adversary. In this case the attempt is deemed to be unacceptable, and a Register EA Done, Unacceptable (RADu) event is issued.

NOTE WELL: additional Endpoint Address protocol operations (such as registering additional addresses or retracting addresses) may occur, but only after the REGISTER EA state has terminated successfully.

Register EA Silent The Register EA Silent state is entered only as a result of a previous state terminating unacceptably (*e.g.*, HELLOING terminated with a Helloing Done, Unacceptable (HDu) event). In this case, the register-EA operations are allowed to proceed, but the node MUST NOT register those addresses with the ZPR network. When the register EA finishes, a “Register EA Done” is issued (regardless of the reason) and ACTIVE, Silent is entered, black-holing the peer’s traffic.

Error Some kind of error that is not automatically recoverable has occurred.

3.3.2. Events

Close A request has been made (*e.g.*, via a local management operation or due to a Policy directive) to terminate the connection. The close operation is then performed (Terminate Request and Response messages are exchanged).

Any Endpoint Data packets received for or from a Link or Docking Session against which a CLOSE event has occurred are silently discarded. These are not considered errors.

Close Complete When the close completes (that is, either the Terminate Indication has been sent, or the Terminate Request and Response have been exchanged, allocated resources have been reclaimed, etc) then the “CLOSE COMPLETE” event occurs.

Note that with the close, there is no “error” condition – if an error occurs then the close is still deemed to be successful.

Config Done All configuration information necessary to bring up a Link or Docking Session has been received. This includes, but is not limited to, the Substrate Address, the Substrate Address of the other end of the Link, cryptographic information necessary to establish the secure link, Policies dictating when the Link or Docking Session should be brought up, and so on.

When CONFIG DONE is issued:

1) If dynamic keying is to be used, SA 0 (the “Null SA”) MUST be available for the keying protocol to use,
2) If predistributed keys are used, then the SAs for those keys MUST be available.

If the Link’s or Docking Session’s configuration is inconsistent, incomplete, or otherwise unacceptable (or unworkable) then the CONFIG_DONE MUST NOT occur. An implementation MUST either:

- Issue an ERROR event (going to the ERROR state). This might indicate some kind of unrecoverable error in the configuration.
- Remain in INITIAL state. This might be used if, for example, more configuration data is required and can be reasonably expected.

A CONFIG DONE in any state other than INITIAL is explicitly defined as a NOOP (that is, the state does not change and no actions are taken). The reason is that the administrators may not wish to place in service the configuration changes that caused the CONFIG DONE event to occur. The administrators must explicitly place the changes into service by restarting the Link or Docking Session.

Error This event can occur in any state and is some form of error or other condition that prevents the Link from operating in the normal manner AND from which automated recovery (*e.g.*, “retransmitting a packet”) is not possible.

In all cases, this results in an immediate transition to the ERROR state.

Given that ERROR events may indicate internal problems that may preclude generating packets, a node MAY send a Terminate Indication (§ 4.3.3) but MUST NOT rely on the peer receiving the indication.

In all states, if an ERROR event occurs then the error MUST be reported to the error reporting system. The error reporting system is responsible for error flow control and rate limiting, etc.

The only way to exit the ERROR state is via a RESET.

Helloing Done, Error This event indicates that an error (other than a security or policy violation, below) occurred in attempting to perform a Hello.

Errors in this category are things like - Lack of resources, - Network communications failures, - Peer not responding after designated number of retransmissions, - Etc.

In short, as opposed to the “unacceptable” case, below, the occurrence and reporting of these conditions do not reveal any sensitive policy or configuration information.

A Terminate Indication indicating the error (§ 4.3.3) SHOULD be sent.

Helloing Done, OK This indicates that the hello protocol has successfully exchanged the relevant hello information.

When a Link has sent a HELLO REQUEST and received a matching HELLO RESPONSE AND has received a HELLO REQUEST and sent an appropriate HELLO RESPONSE then the HELLO DONE event occurs.

Helloing Done, Unacceptable The info in the hello is not acceptable and revealing this to the peer would expose sensitive policy information to a potential attacker. An HDu is issued in this case.

The peer MUST NOT send a Terminate Indication or Request or otherwise inform the unacceptable peer of the situation. The rationale for this is that the unacceptable peer may be attempting some kind of attack (such as scanning) and getting an indication back may leak information regarding the ZPR Networks policies, configuration, and so on.

Keying Done, Error This event indicates that some kind of error (other than a security or policy violation) occurred in attempting to establish keys.

Errors in this category are things like - Lack of resources, - Network communications failures, - Peer not responding after designated number of retransmissions, - Etc.

In short, as opposed to the “unacceptable” case, the occurrence and reporting of these conditions do not reveal any sensitive policy or configuration information.

If this event occurs in the INITIAL or INACTIVE states it SHOULD be treated as a programming error since it cannot happen (the keying operations are not started until the START event transitions from INACTIVE to KEYING). The next state is ERROR.

If this occurs in the ERROR or CLOSING states it is a NOOP.

If it occurs in any other state, it is treated as an error; start the process of closing the Link or Docking Session and transition to the CLOSING state.

Indicating the error to the peer is left to the keying protocols because sending a Terminate Indication would be unsecure.

Keying Done, OK This event occurs when the Link is in the KEYING state and enough cryptographic material is available for the Link to use session keys for all traffic.

If session keys are pre-distributed, this event automatically occurs after entering the KEYING state.

If a keying protocol is in use, then it occurs when the protocol has created session keys and installed them in the Link. This typically occurs after several packets have been exchanged between the two end points of the link.

This event occurs when the Link is in the KEYING state and enough cryptographic material is available for the Link to use session keys for all traffic.

If session keys are pre-distributed, this event automatically occurs after entering the KEYING state.

If a keying protocol is in use, then it occurs when the protocol has created session keys and installed them in the Link. This typically occurs after several packets have been exchanged between the two end points of the link.

If this event occurs in the INITIAL or INACTIVE states it SHOULD be treated as a programming error since it cannot happen (the keying operations are not started until the START event transitions from INACTIVE to KEYING). The next state is ERROR.

If this occurs in the ERROR or CLOSING states it is a NOOP.

In any other state, this event is treated as a NOOP. The assumption is that the keying protocol is running, performing whatever operations it normally performs, and these operations are completing without error (*e.g.*, updating session keys). This has no effect on the FSM.

Keying Done, Unacceptable If a security violation is detected (such as the peer fails some authentication check, a certificate is presented with an unknown or unacceptable signer, etc) then the keying terminates with an “unacceptable” condition.

The peer MUST NOT send a Terminate Indication or Request or otherwise inform the unacceptable peer of the situation. The rationale for this is that the unacceptable peer may be attempting some kind of attack (such as scanning) and getting an indication back may leak information regarding the ZPR Networks policies, configuration, and so on.

If this event occurs in the INITIAL or INACTIVE states it SHOULD be treated as a programming error since it cannot happen (the keying operations are not started until the START event transitions from INACTIVE to KEYING). The next state is ERROR.

If this occurs in the ERROR or CLOSING states it is a NOOP.

Keepalive Failure This event occurs if A) A keepalive is enabled on a Link or Docking Session and B) the keepalive is deemed to have failed. If keepalive is not enabled then this event cannot occur.

Reset The RESET event is an event that may be used in error recovery. If issued, the Link or Docking Session should be restored to its power-on state. The new FSM state is to INITIAL – the Link or Docking Session is now in a condition to receive a new configuration and attempt to start up. The notion is that this clears any bad accumulated state, restoring the Link or Docking Session to a known good condition.

If not in any of the silent states (e.g., HELLOING SILENT) or a state where it is impossible to send a packet *(e.g.,* ERROR or INITIAL state) then the node SHOULD send a Terminate Indication to the peer indicating that the Link has been reset.

RESET MAY be used to shut down a Link or Docking Session as a part of the process of shutting down a Node.

All resources obtained for the Link or Docking Session MUST be reclaimed.

Register EA Done, Error Some error has occurred while registering an Endpoint Address. These errors are such that disclosure of them to a potentially compromised peer would not leak sensitive information. Such errors include running out of storage, network transmit or receive issues, or too many retransmissions.

A Terminate Indication indicating the error (§ 4.3.3) SHOULD be sent and the Docking Session closed.

Register EA Done, OK This event occurs only on Docks and Adapters, with respect to a Docking Session.

The Adapter has successfully registered at least one Endpoint Address with the Dock.

The Dock distributes a binding of the Endpoint Address and the Dock Session Address to the rest of the ZPR Network, enabling Endpoint Packets addressed to the Endpoint to be forwarded to the correct Dock. This process is done by the Admin Service.

Register EA Done, Unacceptable An anomalous situation has occurred. This situation is such that disclosure of the situation to a potentially compromised peer would leak sensitive information. Such errors include attempting to register an unapproved Endpoint Address.

Receive Hello Message This event occurs only on a Dock with regard to a Docking Session that is in the LISTENING state.

This event indicates that a Hello management message has been received from an Adapter. The message is passed to the hello protocol and the protocol started.

This message is valid only if predistributed keys are used. If not, it is discarded and an error occurs.

In effect, this means that an Adapter is attempting to “sign on” to the ZPR Network.

Receive Keying Message This event occurs only on a Dock with regard to a Docking Session that is in the LISTENING state.

This event indicates that a keying message has been received from an Adapter. The message is passed to the keying protocol and the protocol started.

This event is valid only if predistributed keys are not used. If not, it is discarded, and an error event occurs.

In effect, this means that an Adapter is attempting to “sign on” to the ZPR Network.

Start Start is an external event, driven by a management process. It occurs when that process decides that a Link should be started. *Typically,* this is driven by a set of ZPR Policies that define when a link should be started. Note that one such policy may be “always bring up the Link”.

If a START event occurs in any state other than INACTIVE, it MUST be treated as NOOP. It MAY be reported as a programming or execution error via the logging system.

Event/State combinations not shown in Figs. 3.2, 3.3, 3.4 are treated as no-ops. They do not cause state transitions nor do they initiate any processing or protocol operations. They MAY be logged and/or counted.

The following tables show the complete FSM:

Event	**Initial**	**Inactive** Link/Adap	Dock	**Listen**	**Closing**	**Error**
Config Done	Inactive	–	–	–	–	–
Start	–	H/K[1]	Listen	–	–	–
Keying Done$_{\text{Any}}$	Error	Error	Error	Error	–	–
Helloing Done$_{\text{Any}}$	Error	Error	Error	Error	–	–
RegisterEA Done$_{\text{Any}}$	Error	Error	Error	Error	–	–
Rx Keying Msg	Error	Error	Error	Keying	–	–
Rx Hello Msg	Error	Error	Error	Helloing	–	–
Close	–	Closing	Closing	Closing	–	–
Close Complete	Error	Error	Error	Error	Inactive	–
Reset	Initial	Initial	Initial	Initial	Initial	Initial
Error	Error	Error	Error	Error	Error	–
Keepalive Failure	–	–	–	–	–	–

Table 3.1: Full Link/Docking Session FSM (unconnected states)

Event	**Keying**	**Helloing** Link	Dock/Adap	**RegisterEA**	**Active** [2]
Config Done	–	–	–	–	–
Start	–	–	–	–	–
Keying Done$_{\text{O}}$	Helloing	–	–	–	–
Keying Done$_{\text{U}}$[3]	Helloing$_{\text{S}}$	Helloing$_{\text{S}}$	Helloing$_{\text{S}}$	RegisterEA$_{\text{S}}$	Active$_{\text{S}}$
Keying Done$_{\text{E}}$	Closing	Closing	Closing	Closing	Closing
Helloing Done$_{\text{O}}$	Error	Active	RegisterEA	–	–
Helloing Done$_{\text{U}}$	Error	Active$_{\text{S}}$	RegisterEA$_{\text{S}}$	RegisterEA$_{\text{S}}$	Active$_{\text{S}}$
Helloing Done$_{\text{E}}$	Error	Closing	Closing	Closing	Closing
RegisterEA Done$_{\text{O}}$	Error	Error	Error	Active	–
RegisterEA Done$_{\text{U}}$	Error	Error	Error	Active$_{\text{S}}$[4]	Active$_{\text{S}}$
RegisterEA Done$_{\text{E}}$	Error	Error	Error	Closing	Closing
Rx Keying Msg	–	–	–	–	–
Rx Hello Msg	Error	–	–	–	–
Close	Closing	Closing	Closing	Closing	Closing
Close Complete	Error	Error	Error	Error	Error
Reset	Initial	Initial	Initial	Initial	Initial
Error	Error	Error	Error	Error	Error
Keepalive Failure	–	–	–	–	Closing

[1]Either Helloing or Keying is done, depending on whether predistributed keys or a keying protocol are used, respectively.

[3]If the transition to a SILENT form occurs due to an unacceptable KEY DONE event from a state other than KEYING, then the transition occurs while a protocol is executing. The protocol's state MUST be carried over to the new state so that the protocol may continue "as if nothing had happened".

[4]Internal state remains the same, but the Link or Dock Session "goes silent".

Event	**Keying**	**Helloing**		**RegisterEA**	**Active** [2]
		Link	Dock/Adap		

Table 3.2: Full Link/Docking Session FSM (operational states)

Event	**Helloing$_S$**		**RegisterEA$_S$**	**Active$_S$**
	Link	Dock/Adap		
Config Done	–	–	–	–
Start	–	–	–	–
Keying Done$_{O/U}$	–	–	–	–
Keying Done$_E$	Closing	Closing	Closing	Closing
Helloing Done$_{O/U}$	Active$_S$	RegisterEA$_S$	–	–
Helloing Done$_E$	Closing	Closing	Closing	Closing
RegisterEA Done$_{O/U}$	Error	Error	Active$_S$	–
RegisterEA Done$_E$	Error	Error	Closing	Closing
Rx Keying Msg	–	–	–	–
Rx Hello Msg	–	–	–	–
Close	Closing	Closing	Closing	Closing
Close Complete	Error	Error	Error	Error
Reset	Initial	Initial	Initial	Initial
Error	Error	Error	Error	Error
Keepalive Failure	–	–	–	Closing

Table 3.3: Full Link/Docking Session FSM (silent states)

More details on the various transitions and procedures are presented in following sections.

3.4. Initial Configuration Data

Each of the following sections describe the initial configuration[5] data required in order to perform the function outlined in the section.

Specification of how these entities are configured is beyond the scope of this note. Techniques include, but are not limited to,

- Manual entry of configuration data
- Obtaining data via a configuration protocol such as DHCP
- Distribution via some medium (such as a USB memory device)

[2]Changing from ACTIVE to ACTIVE, SILENT, the internal state of the Dock Session or Link remains the same EXCEPT that it “goes silent”. Received Endpoint Data Packets and Endpoint Data Packets to be transmitted are silently discarded.

[5]In this context we mean configuration in the sense of the data needed to operate, such as addresses, initial keying material, and so on. We are *not* referring to the ZPR Configuration.

Regardless of the mechanism(s) used, the information is considered to be valid and is trusted.

Configuration information may be changed during normal operations. For example, in the course of adding a new Forwarder to a ZPR Network, the Admin Service could distribute new Link configurations to one or more existing Forwarders, thus configuring the existing Forwarders to establish Links to the new Forwarder.

3.5. Link Establishment

Links are created between Forwarders. A Link must be created and started prior to traffic flowing between the Forwarders.

3.5.1. Required Configuration Data

Prior to attempting to establish a Link, the two Nodes at each end of the Link must be configured[6] with the following information:

1. The local Node's Node Address.
2. The Node Address of the expected peer.
3. Cryptographic configuration, which is one of:
   a. Necessary security associations (session keys, etc.) distributed in a secure manner by the Admin Service prior to Link establishment. These SAs are distributed to both sides of the Link. The active SA is specified as a part of this process.
   b. Cryptographic material necessary to bring up a security association with the peer using a key management protocol. This includes information needed to cryptographically identity the Node being configured with the peer (*e.g.* an X.509 Certificate for the Node, with asymmetric keying material), identify the peer to this node, and so on. This information is used by a dynamic keying protocol (*e.g.*, IKEv2) to establish active SAs as a part of bringing up the Link.
4. Policies describing the conditions under which it is acceptable to establish the Link. These Policies are distributed as Link Connection Policies. Note that a valid Policy can be "always".

   If the Policy indicates that the Link should not be established then the Policy SHOULD indicate whether any attempt to do so be rejected as an error or as unacceptable (that is, the keying state is terminated with Keying Done, Error or Keying Done, Unacceptable). The default is to terminate as unacceptable.
5. For IP substrates, the Substrate Address, IP Protocol and port number (as appropriate) in order to contact the peer.
6. Any additional attributes or configuration information that may apply to the Link, such as:
   a. Expected lifetime of the Link,
   b. Explicit conditions to terminate the Link,
   c. Maximum amount of transmit bandwidth to use on the Link,

[6]Configuration mechanisms are out of scope for this document.

d. Maximum transmit queue length for the Link,
e. Maximum receive rate on the Link (*e.g.* receive traffic policer
f. Any Substrate-specific QoS attributes to apply (such as DSCP for IP substrates).

3.5.2. Procedure

The procedure is:

1. If the SAs are not preconfigured by the Admin Service, a dynamic keying protocol (*e.g.,* IKEv2) is used to generate SAs.

   If the keying protocol requires that one side start the key negotiation, then the node with the numerically smaller Node Address (treating the addresses as unsigned integers) initiates keying. When conditions are appropriate, it starts the key agreement protocol. The other Node always waits. The waiting Node MUST monitor conditions and if they are such that the Link ought to be initiated, but is not, the waiting Node SHOULD make a Report to the Admin Service.

   If the keying protocol allows either side to start the operation, then either may do so. The peer evaluates conditions and accepts or rejects the keying accordingly.

2. After the key protocol finishes and security associations have been set up, the Nodes exchange Hello Request and Response Management Messages (§ 4.3.4) with each other. Each Node sends a Hello Request and, in turn, expects to receive a Hello Response. Similarly, when a Node receives a Hello Request from the peer, it formulates and sends an appropriate Hello Response. If a Response does not indicate success, the Link is torn down.

3. Each Node generates a Link Address and assigns it to the Link.

4. Each Node starts Link monitoring via ZDP Echo Request and Response Management Messages (§ 4.3.2). Each peer periodically sends an Echo Request. If, per the request/response semantics described in § 4.4, the Echo Request fails[7], then the Link or peer Node is deemed to have failed and the Link transitions to the closing state. While the request/response semantics make accommodation for temporary failures by retransmitting several times, an implementation MAY extend this by automatically attempting to restart the Link by issuing a START event after the CLOSE COMPLETE event and transition to the INACTIVE state. Automatic restart MAY be controlled by the Connection Policy.

5. The topology management functions of the Admin Service on the Node are informed that the Link is available to transport Traffic Flows.

At any point in the process of bringing up the Link, either Forwarder may terminate the process if any acceptability criteria are not met. Acceptability criteria are distributed via the Connection Policy controlling the Link. The Link Termination process (§ 3.18) is used to terminate the Link Establishment.

If at any point in the process, a Forwarder decides that it cannot continue and that the process of bringing up the Link has failed, the Link Termination process (§ 3.18) is used to terminate the Link Establishment.

[7] § 4.4 states that a request is transmitted three times, with a 1-second timeout for each transmission. Thus, the link must be down for 3 seconds before it is declared down.

3.6. Dock Session Establishment

A Dock Session is established between an Adapter and a Dock. The Dock Session must be established prior to establishing individual Tethers for Traffic Flows.

3.6.1. Required Configuration Data

The following configuration data must be in the Dock and Adapter prior to bringing up a Docking Session:

1) An Adapter is configured with the

   a. Substrate Address (and possibly IP protocol and port numbers) of the Dock with which to establish the Dock Session,

   b. Cryptographic configuration, which is one of:

      i. If predistributed keys are used, the necessary security associations (session keys, etc,) distributed in a secure manner by the admin services prior to Link establishment. These SAs are distributed to both sides of the Dock Session. The active SA is specified as a part of this process.

      ii. If a keying protocol is used, the cryptographic materials needed by the keying protocol (*e.g.*, IKEv2) to establish a Security Association with the Dock. This material may include X.509 certificates for both the Adapter and the Dock.

   c. Specification of the conditions under which the Adapter should establish the Dock Session. This may be included as a part of the Connection Policies governing the behavior of the Adapter.

   d. Whether the Adapter should register the Endpoint's Endpoint Address with the Dock and, if so, what address to register (or to use whatever address the Endpoint has been assigned by, *e.g.*, DHCP).

      Ed. Note: I believe that this is necessary to handle the case where the Adapter/Endpoint gets an IP address from the substrate's DHCP service. We cannot know what that address is/will be but (I think) have to use it…

   An Adapter may be configured with several different Docks with which to (attempt) to establish Dock Sessions (including the cryptographic material) and different conditions for establishing Dock Sessions. How the Adapter chooses a Dock out of this set, whether it goes from one to the next in the event of failure, and so on, are local implementation decisions.

2) A Dock is configured with

   a. Cryptographic configuration, which is one of

      i. Necessary security associations (session keys, etc.) distributed in a secure manner by the admin services prior to Link establishment. These SAs are distributed to both sides of the Dock Session. The active SA is specified as a part of this process.

ii. Cryptographic materials needed by a dynamic keying protocol (*e.g.*, IKEv2) to establish a Security Association with the Adapter. This material may include X.509 certificates for both the Adapter and the Dock.

b. Specification of the conditions under which it will allow the Adapter to initiate a Dock Session. This may be included as a part of the Connection Policies governing the behavior in the Dock.

c. Whether the Dock should accept a Register Endpoint Address message from the Adapter or not and, if so, what addresses to accept ("all addresses" is allowed).

This configuration information is distributed by Docking Connection Policies.

3.6.2. Procedure

The following steps are taken to establish a Dock Session:

1) The Adapter continuously monitors various parameters of its environment. When it detects the correct conditions to establish a Dock Session to a Dock or Docks, it initiates the Dock Session.

   a. If there are multiple sets of conditions that have been met, the Adapter selects one in an implementation-dependent manner.

   b. If a given set of conditions specifies multiple Docks, the Adapter selects one in an implementation-dependent manner.

   Dock Sessions are always initiated by the Adapter to improve reliability and scaling, as well is to minimize probing:

   - The platform executing the Dock's functionality is likely to serve many Docks and Dock Sessions. Constantly probing to determine whether an Adapter was up and running would consume a lot of resources and generate a lot of useless network traffic.
   - In many cases, the platform on which the Endpoint (and probably Adapter) executes will not have a fixed Substrate IP address. It would get it via some DHCP-like mechanism. It would be impossible for the Dock to know this address in advance.

2) If an SA has not been pre-distributed then the Adapter initiates an IKEv2 key exchange with the selected Dock. If the key exchange fails, for any reason, then the Adapter MAY attempt a key exchange with another Dock (if it has multiple Docks for the initiation conditions). As a part of doing the key exchange, the Dock or Adapter may terminate the operation if the peer's identity is not acceptable for some reason. Acceptability criteria are distributed via Docking Connection Policies.

   In order to minimize the effect of key exchange flooding attacks, a Dock's configuration includes token-bucket based rate-limiters and other filters. The Dock evaluates the initiation of the key exchange against these rate limiters and filters and continues with the exchange if and only if the exchange is permitted by them. Key exchange attempts that fail to pass the rate limiters and other filters MUST be reported as they represent potential attacks or security issues.

RFC 8019[8] specifies mechanisms to prevent IKEv2 responders from succumbing to DDOS attacks from attackers masquerading as legitimate IKEv2 initiators. All ZPR entities (Docks, Links, Adapters, etc.) MUST implement these mechanisms in order to protect a ZPR network from attackers masquerading as ZPR entities.

An attempt at initiating a Dock Session that is not accepted because it violates rate limiters or other filters MUST be reported as it represents a potential security issue.

> **Ed. note**: It seems clear that there should be some controls on creating Docking Sessions. A rate limiting function is needed to minimize flooding attacks. I think that there should be additional filters (*e.g.,* Dock *foo* accepts sessions only from Adapters/Endpoint Endpoint *bar*, *baz*, and *bozo*). These filters may even have further qualifications (*e.g.* Endpoint 86 might be allowed to connect from 8am to 5pm) and have per-Endpoint rate limits. This should be worked out in more detail.

3) After the key exchange completes and a security association has been set up, the Adapter and Dock exchange Hello Request and Response Management Messages (§ 4.3.4) with each other. The Hello Response may terminate the Dock Session, accept the Dock Session, or redirect the Dock Session to another Dock. The Adapter tells the Dock the name of the policy by which the Adapter initiated the Dock Session. In return the Dock tells the Adapter the name of the policy under which it accepted the Session.

4) The Adapter MAY inform the Dock of the Endpoint Application's Endpoint-Address(es) via Register Endpoint Address Management Messages (§ 4.3.8). If the address is not acceptable to the Dock then it is rejected and the Dock Session is terminated. The Adapter does this if

   a. Its Docking Connection Policy indicates that it is serving an Endpoint that is a server and the network needs to know the Server Endpoint's Endpoint Address so that initial requests from clients can be properly routed to the server or

   b. If the Docking Connection Policy indicates that the Endpoint's Endpoint Address may not be known or predictable ahead of time (*e.g.*, an Endpoint for a travelling employee of a company who can connect to a company's ZPR network from remote locations such as airports or hotels).

5) The Dock generates and assigns a Dock Session Address to each registered Endpoint Address. The Dock generates a set of Dock Session Address/Endpoint Address mappings and passes them to the Admin Service for distribution to the rest of the network. The Dock Session Address is added to the topology management system so that potential peers can send packets to the Endpoint.

At any point in the process of bringing up the Dock Session, either the Dock or the Adapter may terminate the process if any acceptability criteria are not met. Acceptability criteria are distributed via Docking Connection Policy.

After the Dock Session is established, each side starts sending Echo Request and Response Management Messages (§ 4.3.2) to the other to monitor the health of the Session. The parameters of the keep alive are defined in the Docking Connection Policy.

[8] Nir and Smyslov (2016)

Note Well: After a Dock Session has been successfully brought up, the Endpoint Address and Dock Session Address by which the Endpoint Address may be reached has been distributed throughout the ZPR Network.

If a Dock Session abnormally terminates (*e.g.,* due to a keep-alive failure) then it is up to the Adapter to decide whether to reestablish the Dock Session or not. It performs the evaluation and executes the decision-making process described in step 1) above.

If at any point in the process, a Dock or Adapter decides that it cannot continue and that the process of bringing up the Docking Session has failed, the Dock Session Termination process (§ 3.16) is executed.

The case of an Adapter having multiple Dock Sessions is left for later work.

3.7. Adapter to Dock Tether Establishment and Endpoint Packet Transmission

Tethers are established under two circumstances; first, when an Adapter receives a packet from the Endpoint for a potentially new Flow (*e.g.*, when a new TCP connection starts to be established) and second, when a Dock receives a packet from the ZPR Network for a new Flow to an Adapter. This section covers the first case, where an Adapter attempts to bring up a new Tether. Section 3.8 covers the case of a Dock initiating a Tether with an Adapter.

The Adapter maintains an Endpoint Lookup Table (ELT) of all active Flows being transmitted into the ZPR network (that is, Flows which have been established, for which a Visa has been granted and installed in the network, and for which a Stream has been provisioned and a Stream ID obtained). The table's key is a Traffic Classification Specification (for example, the Endpoint Packet's source and destination IP addresses, IP protocol, and possibly the source and destination transport (UDP or TCP) ports). If the key is not found in the table, a "not found" indication is produced. If the key is found, a PEP is produced that describes how to transmit the packet (which Dock to send it to, what Stream ID to use, etc.).

When the Adapter receives an IP packet from the Endpoint, it looks the packet up in the ELT.

1) If the ELT lookup produces a PEP then the directives of the PEP are followed. Typically, the Endpoint Packet's IP header is compressed (per the PEP, § 3.25), the Endpoint Packet is encapsulated in a ZDP Transit Packet (§ 4.2.5) and the packet is sent to the Dock and with the Stream ID specified in the PEP.

2) If the ELT lookup does not produce a PEP then it indicates that a new Stream is to be created:

    a. A new entry is added to the Endpoint Lookup Table. The Traffic Classification Specification for the entry is the source and destination addresses, etc., of the Endpoint Packet. The PEP for this entry specifies that the Endpoint Packets should be buffered pending the completion of the setup.

       While the registration is in progress, the PEP in the Endpoint Lookup Table entry indicates that fact, preventing additional lookups, etc. If any further Endpoint Packets arrive at the Adapter for the entry being constructed, only the most recently received packet should be buffered.

b. A Bind Endpoint Addresses request (§ 4.3.9) is sent to the Dock requesting a Stream ID to use when sending Endpoint Packets with the given addresses, protocol, and ports, to the Dock.

c. The Dock evaluates the request and may take one of three actions:

   i. It may initiate the ZPR Authentication Process (§ 3.26). If the ZPR Authentication process fails, the request is rejected. The Dock sends a Bind Endpoint Addresses Response with an appropriate error code. If the process succeeds, then the Visa Service has installed a Visa in the network (including the Dock) and appropriate Streams provisioned. The Dock then sends a Bind Address Response to the Adapter, specifying a Traffic Classification Specification and a Stream ID to use when sending traffic that matches the filter.

   ii. It may immediately reject the request. If so, the binding fails and the Dock sends a Bind Endpoint Addresses Response with an appropriate error code.

   iii. It may immediately accept the request. If the request matches a Traffic Classification Specification that is installed in the Dock (and, presumably, for which a Visa has been installed) then the Dock may immediately accept the request. The dock sends a Bind response indicating success and with a Stream ID and Traffic Classification Specification for the Adapter to use.

   This covers the case where a Visa's Traffic Classification Specification includes wildcards, ranges, or the like, but the Traffic Classification Specification given to the Adapter is only for a specific set of values. For example, if a Traffic Classification Specification is

```
source=1.2.3.4, dest=5.6.7.8,
protocol=UDP,
source-port=any, dest-port=53,

use streamed=98765
```

   then the Dock may give explicit Traffic Classification Specification to an adapter for each explicit 5-tuple that matches.

```
Bind Request: 1.2.3.4, 5,6,7,8, UDP, 11111, 53
Bind Response filter=1.2.3.4, 5,6,7,8, UDP, 11111, 65
Use Stream ID 98765

Bind Request: 1.2.3.4, 5,6,7,8, UDP, 22222, 53
Bind Response filter=1.2.3.4, 5,6,7,8, UDP, 22222, 65
Use Stream ID 98765
```

   Etc.

   The intent of the Stream's Traffic Classification Specification is that traffic from the Endpoint to the DNS server (UDP Port 53) at 5.6.7.8 from any source port at the Endpoint is to be allowed. This is required because the source-port in these cases is randomly chosen. The Dock could give the entire Traffic Classification

Specification, including the wildcard, to the Adapter, but that might reveal policy details.

The dock SHOULD set DFT forwarding rules for the explicit 5-tuples that have been formally requested and admitted (that is, in the above example, there should be forwarding entries for 1.2.3.4:1111->5.6.7.8:53 and 1.2.3.4:22222->5.6.7.8:53 and NOT for 1.2.3.4:*->5.6.7.8:53.

d. If the Dock rejects the request, then it SHOULD log that fact.

e. If the Adapter receives a response rejecting the request it SHOULD log the request. The log entry should include all available diagnostic information, especially the reason code and additional status information in the Response message.

   The Adapter SHOULD also send an IP "ICMP Destination Unreachable, administratively prohibited" message to the Endpoint.

f. If the Dock accepts the Binding request it generates a successful Bind Response. The Adapter receives the response and finalizes the DLT entry – primarily updating the DLT entry and PEP with the response's Stream ID.

g. Any buffered Endpoint Packets are then encapsulated in Transit Packets (§ 4.2.5) and transmitted to the dock in the normal manner.

This procedure only sets up the Adapter-to-Dock Stream on the Tether. It does not set up the reverse flow; it is set up by the Dock when it has packets to send to the Adapter. There are two reasons for this: 1) For some applications, there might not be a reverse flow so setting one up would waste resources, and 2) having the Dock set up the reverse flow of the Tether when the Dock has packets to send to the Adapter allows for the possibility that there are different Docks for each flow.

On receiving an Endpoint Address binding, the Dock

1) Evaluates the addresses for acceptability. If the addresses are unacceptable for reasons other than policies, it formulates the correct response and sends it to the Adapter.

   If the addresses are unacceptable for policy reasons, it creates a dummy DFT entry that will silently discard packets received on the flow. It reports "success" to the Adapter.

2) If the Dock evaluates the addresses as acceptable it creates an entry in the DFT for the addresses and Stream ID specified in the registration request. The PEP is configured to buffer a single packet and is marked as "needing authentication".

3) The Dock sends a Bind Response indicating success.

4) When the Dock receives a ZDP Transit Packet it looks up the packet's Stream ID in the DFT. If no entry is found for the Stream ID the Dock discards the packet and reports it, as it potentially is an attack (*e.g.,* scanning the Stream ID range to see "what gets through"). The Dock DOES NOT send an error of any kind (a notional "invalid stream ID") message as that would give an attacker useful information. This silence is not a problem for Adapters that are not attackers; if the Adapter made an "innocent error" then the operation will eventually time out and the Adapter will perform some kind of error recovery.

5) If the Stream ID is known, then the lookup will result in a PEP. If the PEP is “Needing Authentication” then the packet is sent to the local Authentication Service for ZPR Authentication. While the PEP is in the Needing Authentication state, it accepts packets for the Stream. The first packet is buffered, the rest are discarded. If authentication completes successfully then a Visa will have been installed in the network – the buffered packet, and all subsequent packets, can be forwarded in the normal manner (§ 3.14).

   If the authentication fails then the packet is discarded and the Tether is terminated (§ 3.9).

3.8. Dock to Adapter Tether Establishment and Endpoint Packet Transmission

A Dock establishes a Tether with an Adapter when it receives a Visa. It provisions each Stream specified in the Visa. Each Stream includes a Traffic Specification for the flow (Endpoint Addresses, IP protocol, and transport ports). This allows the Dock to determine on which Docking Session to send packets for the flow. It can do this because when the Docking Session is established, the Endpoint has registered its Endpoint Address with the dock with a Register Endpoint Address message (§ 4.3.8).

The procedure to set up the flow is basically the same as given in § 3.7 except that the roles are reversed: the Dock makes the request and the Adapter evaluates the Request, allocates a Stream ID, and generates the response:

1) When the Dock receives the Visa, it generates a Bind Endpoint Addresses Request (§ 4.3.9) for each stream specified in the Visa. It sends the request to the Adapter.
2) The Adapter evaluates each request. If it is not acceptable, the Adapter sends a response indicating the error and the procedure terminates. If the request is acceptable, the Adapter allocates a Stream ID and sends it to the Dock in the response.
3) The Adapter also sets up its tables so that packets received from the Dock with the allocated Stream ID are decompressed, with the Endpoint Addresses, IP protocol, and port numbers inserted into the resulting Endpoint Packet.
4) The Dock receives the response and updates the PEP used to forward Transit Packets of the flow to the Adapter to insert the Stream ID from the response into the Transit Packets.

If at any point in the process, the Tether cannot be successfully established then the Tether Termination procedure (§ 3.9) is followed.

3.9. Tether Termination

Tethers may be terminated for a number of reasons, such as imminent shutdown of the Dock or Adapter, an unrecoverable error, authentication failure, receipt of a Terminate Request or Indication (§ 4.3.3) and so on. Regardless of the root cause of terminating the Tether, the Dock and Adapter, independently shall:

1) Replace the PEP used to handle Transit Packets received on the Tether with a “quiescent” PEP: one that will silently discard any Transit Packets received on the Tether. Receipt of

these packets is not considered an error or other pathological condition and MUST NOT cause an error or security report to be generated. NOTE WELL: the Stream ID value that was assigned to the Tether is still in use and MUST NOT be reused at this time.

2) A 10 second timer is started.

3) When the timer expires, the forwarding table entry is deleted, the PEP released, and the stream ID is available for reuse.

This procedure ensures that any Endpoint or Transit Packets destined for the Tether and that are legitimately in the network at the time of the Tether's termination can drain from the network.

3.10. Adapter to Adapter Security

ZPR protects Endpoint Packets while in transit through the ZPR Net by using Adapter-to-Adapter Security (A2A). Currently, only message integrity is protected; encryption could be added at a later date.

The cryptographic material (keys, algorithms, parameters, SAIDs, and so on) used by the Ingress and Egress Adapters is generated and distributed by the Visa Service as a part of installing the Visa in the network. The Docks pass this material to the Adapters via the Bind Endpoint Address response message associate this material with the specific Stream being provisioned by the Visa.

When an Ingress Adapter receives a packet from the source Endpoint, the A2A MICV is calculated over the entire IP packet before compressing the packet. That is, the SA, DA, etc. as received from the Endpoint are covered by the MICV.

The SAID and resulting MICV are inserted in the ZDP Transit Packet (§ 4.2.5).

When an egress Adapter receives a ZDP Transit Packet from the ZPR Net it first looks up the packet's Stream ID, producing (among other things) 1) a table of A2A Security Associations and 2) The values of the fields compressed out of the packet by the Ingress Adapter. The A2A SAID in the packet is looked up in this table. The SA includes the cryptographic material necessary to validate the integrity of the Endpoint Packet.

If the A2A SA lookup fails it is considered a potential security issue. The packet is silently discarded, and the event reported as a security issue.

The receiving Adapter decompresses the packet, generating the original Endpoint Packet using the field values looked up in the Stream ID table. It then calculates the MICV and checks it against the MICV in the packet. If the MICV is good, the packet is forwarded per the normal forwarding procedure. If it is not good, the packet is silently discarded and a report, indicating a potential security issue, generated.

Note Well: the Docks and intermediate Forwarders do not have the A2A SA cryptographic material. They do not check the A2A MICV, nor can they generate one. This prevents a compromised Dock or Forwarder from injecting undetected bad Endpoint-to-Endpoint data.

3.11. Visa Installation and Stream ID Request

Visas and their associated Streams are installed in Nodes by the Visa service. The service installs just the information that is required by the Node to perform its function. The Dock passes required information to the Adapter via the Bind Response (§ 4.3.9). Installed information is:

- The Ingress Adapter gets
  - The name of the Visa,
  - Parameters controlling the Visa, such as lifetime,
  - For each Stream that the Visa can provision:
    - The name of the Stream,
    - Parameters controlling the Stream (such as its expiration time),
    - The A2A security association to use (keys, algorithms, etc. to use when transmitting a packet on the flow,
    - A Traffic Filter specification (such as the 5-tuple) to use to identify traffic for the flow,
    - Parameters describing how the Endpoint is to compress the packet, and
    - Any additional information needed to forward the packet (*e.g.*, QOS parameters).
- The Egress Adapter gets
  - The name of the Visa,
  - Parameters controlling the Visa, such as lifetime,
  - For each Stream that the Visa can provision:
    - The name of the Stream,
    - Parameters controlling the Stream (such as its expiration time),
    - The A2A security association parameters used to process the packet, and
    - The values to insert into fields that are compressed out when compressing transit packets.
- The Nodes get
  - The name of the Visa,
  - Parameters controlling the Visa, such as lifetime,
  - For each Stream that the Visa can provision:
    - The name of the Stream,
    - Parameters controlling the Stream (such as its expiration time),
    - The identity of the next hop to forward the packet to (this is different for each Node), and
    - Any additional information needed to forward the packet (*e.g.*, QOS parameters).

The Stream ID Request process is used to obtain a Stream ID to use when sending a flow via a specific Stream and Visa. In effect, each node in the path asks the next hop “what stream ID shall I use to transmit packets via Visa named X and stream named Y?”

Generally, this process is initiated when a Visa is installed on the Node and the Stream(s) provisioned. The Node sends a Stream ID Request (§ 4.3.5) to the next hop identified in the Stream. If the Visa and Stream have been installed in the next hop, the next hop allocates a Stream ID and returns it in a Stream ID Response (§ 4.3.6), with the “success” status code. If the Visa or Stream has not been installed, the response indicates failure with the “no such name” status code.

If a “no such name” status is received by the upstream, it waits a short amount of time (3 seconds is the suggested value) and then retries the request. It SHOULD retry the request at least 3 times before giving up. The theory is that the downstream might not have received the Visa yet, so this allows the Visa to be received and installed. A node MUST NOT remove a Visa or Stream if all retries of the Stream ID Request fail. The Visa MUST remain installed in the Node until it is removed by normal processes (such as timing out or being explicitly retracted by the Visa Service). Whenever the node receives a packet for the flow it MUST reattempt the heralding process.

While the process is underway a Node MAY buffer the most recent packet received from its upstream for the flow. It SHOULD NOT buffer more than the most recent packet.

Stream setup is unidirectional.

These detailed steps are repeated between each Upstream and Downstream:

1) Each Node receives the Visa from the Visa Service. The Visa includes, among other things:
    a. A Name, assigned by the Visa Service. The name is the same in all Forwarders and Docks.
    b. One or more Streams, specifying the next hop (Node), Stream ID, Stream Name, and so on. Each Node has a different next-hop. The Egress Dock’s next hop is the Egress Adapter.
2) Each Node sends a Stream ID Request to the next hop specified in the Stream. This request includes the name of the Visa and Stream for a Stream ID is desired, as well as an offered Stream ID.
3) The Stream ID Request/Response follows the common request/response semantics (§ 4.4). If the request fails, then the upstream waits a short amount of time and then retries the operation. Only the Visa Service may remove a Visa from a Node, halting the process to get Stream IDs.
4) The Downstream receives the Request. If the Request is deemed to be invalid then the Downstream rejects the Request, sending a Visa Herald Response (§ 4.3.6) specifying the error to the Upstream and the Flow is not established. If the Downstream deems the Request to be valid then, 1) it evaluates the offered Stream ID and either accepts it or allocates a Stream ID; either way, the Stream ID is sent to the upstream in the Response message, 2) it installs any state, allocates resources, and so on, needed by the Stream,

and 3) sends a Stream ID Response (§ 4.3.6) specifying success to the upstream. The Response includes the Stream ID.

3.11.1. First Endpoint Packet Carriage

At each hop the upstream has the option of including the first Endpoint Packet in the Herald Request message. This can save considerable latency in getting the first packet to the destination and starting the setup of the application connection. An upstream may elect not to include first packets; if so, it drops them.

3.12. Proactive Stream ID Installation

When a Node receives a Visa it MAY elect to proactively install a Visa/Stream ID mapping in the upstream. This saves a rtt when the first Endpoint Packet arrives at the upstream.

It does this by sending a Stream ID Response (§ 4.3.6) with the name of the Visa and the Stream ID the upstream MUST use.

If the Visa has not been installed in the upstream, the upstream MAY silently discard the message or it MAY proactively install a “temporary” binding on the expectation that it will get the visa shortly.

If the upstream discards the visa it will eventually get a First Endpoint Packet which will cause it to request the Stream ID from the downstream.

3.13. Route Generation and Distribution

Routes are paths through the Forwarders to specific Egress Tether Addresses.

Routes are calculated by the Admin Service based on

1) The configured network topology (that is, the set of Forwarders and Links that the network’s administrators have defined), and
2) The current state of the configured topology (that is, accounting for which Links and Forwarders may be down, for whatever reason, at a given moment).

The calculations are constrained by policies configured into the Admin Service.

Routes are not distributed to the Docks and Forwarders. Only the Admin Service knows the routes through the network. Routes are utilized by having a path set up in the network by distributing Visas and Provisioning their Streams.

Docks and Forwarders MUST monitor the health of the Links to which they are connected and the peer at the other end of the Link. If an operational Link or peer goes down this MUST be reported to the Admin Service. Similarly, if a non-operational Link or peer becomes operational, this too MUST be reported to the Admin Service.

> **Ed. Note**: This is meant just to cover the general question of how routes are generated and distributed. This should be moved into a new document that just covers ZPR Routing.

3.14. Forwarding

ZDP Transit Packets are forwarded from a source Dock to a destination Dock via 0 or more intermediate Forwarders. Between any two points of the path, an IP Substrate network may also forward ZDP packets along a path of IP routers; this forwarding is transparent to ZDP forwarding.

ZDP forwarding is solely based on the Stream ID in the ZDP Transit Packet. Packets are forwarded along the path to their ultimate Egress Dock. If the Egress Dock is local to the Node, the packets are passed to that Dock.

Packets with unknown Stream IDs are discarded and a Destination Unreachable Management Message (§ 4.3.13) is sent upstream.

> **Ed. Note:** Should we do this? If the upstream is compromised, it gives the upstream a way to determine what IDs are in use. This could be benign, or it could be some kind of attack – probably want to allow a forwarder to send "a few" destination unreachables, but if "too many", it assumes an attack and goes into "silent mode".

The Link over which a ZDP Transit Packet was received plus the Stream ID in the ZDP header unambiguously determines the output handling of the packet including:

- Whether the packet is to be passed to a Dock on the Node (and if so, which one),
- Which outbound Link to transmit the packet out and the Stream ID to place in the ZDP header when the packet is transmitted,
- Any receive or transmit QOS treatments (such as receive rate limiting or transmit metering).

When a packet is forwarded, the Stream ID is changed to the Stream ID that the next hop requires.

The forwarding algorithm is as follows.

1. A Transit Packet is received by a Forwarder from an upstream Forwarder or Dock. The packet's ZDP header contains a ZPR Parameter Identifier (ZPI) and a Stream ID.
2. The ZPI is extracted and used to locate A) The ZPR Configuration describing the packet's format and B) The security parameters under which the packet has been cryptographically protected. If the ZPI is unknown, then the packet is silently discarded.
3. The ZDP header packet is integrity-checked and decrypted per the security parameters. If it fails the integrity check, then the packet is silently discarded.
4. If the packet type field is not 0 (Transit Packet) then the packet is presumed to be a Management packet and is passed to the control plane for further processing.
5. If the packet type is 0 then it is forwarded as follows:
6. The Stream ID is looked up in the receive Link's lookup table.
   a. If the ID is 0, the packet is passed to the control plane for further processing
   b. If the lookup fails (that is, the ID is not in the lookup table) then the packet is discarded. A Destination Unreachable message is sent upstream (§ 4.3.13).

> **Ed. Note:** Should we send the Destination Unreachable? If the upstream is compromised, it gives the upstream a way to determine what IDs are in use.

7. The lookup produced a PEP which has the data necessary to send the packet along its path. The disposition, along with needed information, could be one of

    a. Pass the packet to the local Dock to send out via a Tether to the Egress Adapter. The lookup results identify the Docking Session and Tether out of which the packet is to be sent. The Tether identifies the Security Association to use when sending packets via the Tether (the Tether must exist since the Egress Dock creates the Tether as a result of receiving the Visa in the Visa heralding process, which must precede the first packet).

    b. Forward the packet to the next hop Forwarder. The lookup results identify the Link via which the packet is to be transmitted and the Stream ID to include in the packet's ZDP header. The Link identifies the Security Association to use when sending packets via the Link. The ZDP header is encrypted and a MICV is generated as directed by this association.

       Regardless of whether the packet is sent out a Docking Session/Tether with a Tether ID to the Egress Adapter or via a Link with a Visa ID to the next hop Forwarder, the process is the same.

8. The packet's size is checked against the MTU to the next hop. If the packet is too large to send, then a ZDP Destination Unreachable (§ 4.3.13) is sent upstream. See § 3.19 for more information.
9. The Stream ID found in the lookup is placed in the ZDP Header's Stream ID field.
10. The packet is encrypted and a MICV is generated per the security association as produced by the lookup.
11. If the lookup produced any special transmit handling treatment (such as output rate limiting) specified by PEPs then that treatment is performed.
12. The packet is transmitted.

3.15. Visa or Stream Deletion

Visas or Streams may be deleted from the ZPR Network for several reasons:

- Retraction by the Visa Service,
- Satisfaction of termination criteria.

In addition, Streams may be removed if the Visa that provisioned the Stream is removed.

Removal of Visas and Streams happens locally, independent of other Nodes. It is the responsibility of the Visa Service to ensure that all Visas or Streams are removed.

TBD

3.15.1. Stream ID Withdrawal

When a Node de-provisions a Stream for any reason it should inform the upstream Node that it is doing so. The goal is to cut off the flow of undeliverable traffic as close to the source as possible. It does this by sending a Stream Withdrawal Request (§ 4.3.7) to the upstream. The upstream marks the Stream as unusable. It silently discards any traffic that it would have sent via the unusable Stream. The upstream may then repeat the process, sending a withdrawal to its upstream, and so on. Eventually this will reach the Ingress Adapter, which will formulate the proper ICMP message to send to the source Endpoint (see § 4.4).

3.15.2. Refusal

During the Stream ID acquisition process, a Node may not be able to satisfy the request for some reason other than not having the Stream named in the request. In this case it refuses the request, setting an appropriate error/non-success code in the Stream Id Response message.

If the upstream receives a Stream Id Response with any code other than success or unknown stream then it should

1) Stop attempts to acquire the stream id and inform the Visa service,
2) Withdraw from its upstream any Stream ID it allocated to use the Stream, and
3) Silently discard any traffic that would be transmitted via the Stream Lifetime Expiration.

Visas and Streams are distributed with an expiration time. When that time passes, a Forwarder MUST remove the Visa or Stream from service. Received ZDP Transit Packets that specify the Visa ID of the expired Visa are treated as errors:

- The packet is counted and discarded.
- A log entry MAY be made.
- A Destination Unreachable message (§ 4.3.13) MAY be sent to the upstream Forwarder.

The node with the expired Stream or Visa does not send a Stream ID Withdrawal message because all Nodes MUST implement Visa expiration times, so they will all delete the Visa at (approximately) the same time.

3.16. Dock Session Termination

A Dock Session may be terminated either by the Adapter (*e.g.*, if the Endpoint or Adapter's host is shutting down or the Adapter has lost reachability with the endpoint), or the Dock (*e.g.*, if the Admin Service administratively terminates the session, the Dock's host computer is shutting down, and so on).

In all cases, the Dock or Adapter that initiates the termination sends a Terminate Request message (§ 4.3.3) to the peer Adapter or Dock, respectively. The peer SHOULD respond with a Terminate Response message.

The Adapter performs any termination/shut-down operations that may be appropriate for the Adapter/Endpoint interface. The nature of this interface is beyond the scope of this note.

The Dock retracts all Visas that have been issued for Endpoints using the terminated Dock Session. For Streams for which the Dock is the Ingress Dock, a Stream ID Withdrawal (§ 4.3.7), with the "Ingress Dock Session Terminated" reason code, is issued for each one.

The Dock reports final traffic and accounting statistics on the Dock Session to the admin service prior to deleting the Dock Session's state.

The Tether Addresses of all Tethers using the Dock Session are withdrawn from service.

3.17. Liveness Detection and Failure Processing

Docks and Adapters use Liveness Detection to verify that the Docking Session and peer are operational. Similarly, Forwarders use it to verify that the Link and peer are operational.

The ZDP Echo request and response (§ 4.3.2) messages are used to perform Liveness Detection for Links and Docking Sessions. Independently, each peer on a Link or Docking Session periodically sends appropriate Echo Requests to the other peer and waits for an appropriate Echo Response. The ZDP Echo protocol follows the common Request/Response semantics (§ 4.4); if they declare that the Echo Request has failed then the Link or Docking Session is declared to be down.

The timeouts and number of retransmissions should be large enough to allow for minor faults in the substrate to recover (*e.g.*, for intra-domain IP routing to reconverge, ARPs to be retransmitted, and the like).

When a Link, Docking Session or peer is declared to be down, the Forwarder or Docking Session detecting the failure:

1) Reports the failure to the Admin Service (which, internally, recalculates network topology, establishes new next hops and so on),
2) Stops transmitting via the failed Link or Docking Session. Traffic to be transmitted out the failed Link or Docking Session is silently discarded.
3) The affected Link or Docking Session is marked as down (that is, resources released, the Security Association deleted, and so on).
4) Eventually the Admin Service will withdraw the visas using the failed path and install new ones using a new, operational, path. These operations follow the normal Visa deletion and installation procedures.

Once the above steps are completed, the peer MUST periodically attempt to restart the Link or Docking Session. When the Link or Docking Session comes up, the Node MUST report this to the appropriate service. The service MAY then recalculate routes and withdraw and distribute new Visas accordingly.

Implementations SHOULD NOT include an explicit delay between declaring a Link, Docking Session, or peer is down (that is, when the Echo has failed) and taking the actions described above. They might do this in the hope that the failure is a transient phenomenon of some sort and that it will soon be rectified. This is discouraged since the Request/Response semantics of the Echo provide suitable delays.

When a Link or Docking Session has restarted after having been terminated due to Liveness Detection, a short quiet time MAY be enforced before declaring the Link or Docking Session up to the Admin Service. Liveness Detection is still started, however. This quiet time is used to damp out oscillations due to flapping interfaces and other similar network phenomena. If supported, the quiet time MUST BE specified by the appropriate policy.

Ed. Note: I am concerned that for large, complex, networks this may place an undue burden on the services. It may also offer an avenue of attack; if an attacker targets an IP router in the substrate, it may be able to cause the router to fail, leading to the ZPR Link failing, etc. Is there some way we can do some rerouting, etc. at the Node level? Could a Visa include instructions for what to do if the out-bound Link fails? (something like "if it fails ... use this other link" or "if it fails, tell the upstream that the stream needs rerouting (recurse)"? **TBD**

3.18. Link Termination

TBD

3.19. MTU Exceeded

When ZDP Transit Packets are forwarded through the network it is possible that they are too big to be transmitted via the Link to the next hop Forwarder (see § 5.2 for more background). This is detected by a Forwarder, Egress Dock, or Egress Adapter when it attempts to send a packet. There are several cases to consider:

1) The Egress Adapter detects the condition (the packet exceeds the MTU of the path between the Adapter and Endpoint).

   The Egress Adapter sends a Destination Unreachable, MTU Exceeded message (§ 4.3.13) to the Egress Dock.

2) The Egress Dock detects the condition: either the packet is larger than the Egress Docking Session MTU or the Egress Dock receives a Destination Unreachable, MTU Exceeded message from the Egress Adapter.

   The Egress Dock sends a Destination Unreachable, MTU Exceeded message (§ 4.3.13) to the upstream Forwarder.

3) A Forwarder Detects the Condition (the packet is larger than the Link MTU or the Forwarder received a Destination Unreachable, MTU Exceeded message from the Egress Dock or an upstream Forwarder.

   The Forwarder sends a Destination Unreachable, MTU Exceeded message (§ 4.3.13) to the upstream Forwarder or Ingress Dock.

4) The Ingress Dock detects the condition.

   The Ingress Dock sends a Destination Unreachable, MTU Exceeded message (§ 4.3.13) to the Ingress Adapter.

5) The Ingress Adapter detects the condition (on the Ingress Adapter to the Ingress Dock Docking Session).

   The Adapter informs the Endpoint via the normal mechanisms for the Adapter/Endpoint interface (*e.g.* sending an ICMP Destination Unreachable message to the Endpoint).

The elements along the path SHOULD NOT remember a new path MTU, based on the MTU exceeded messages. IP's MTU management protocols (both for IPv4 and IPv6) require that the source IP host limit the PMTU per received ICMP messages and then to periodically probe the path to see if the PMTU has increased. It does this by sending larger packets. It is adequate that the IP host do this; there is no need for ZPR to also do it. In fact, if ZPR enforces a smaller PMTU before the bottleneck, it is possible that ZPR's efforts would conflict with the source Endpoint's efforts, leading to pathological behavior.

3.20. Message Receive Limiting

Forwarders, Docks, and Adapters SHOULD implement mechanisms to mitigate flooding denial of service attacks using management messages.

We recommend that mechanisms such as strict input queue limits or receive rate limiting of management messages be used in order to prevent a compromised peer from flooding the control plane of the Dock, Forwarder, or Adapter.

Messages that exceed the limits MUST be reported and silently discarded. The report mechanism MUST NOT, itself, become a target for flooding. It should do something like periodically reporting a summary of the number of messages that exceed the limits.

3.21. Endpoint Address/Tether Address Binding Distribution

When an Adapter learns the Endpoint Address of its Endpoint, it sends a Bind Endpoint Address Request (§ 4.3.9) to its Dock. The Dock assigns a Tether Address to the Endpoint Address and then adds this binding to the Admin Service by informing the local Admin Service Endpoint of the binding.

Bindings are removed from the Admin Service for a number of reasons:

1) Adapter may send a debind message if the Endpoint Address is no longer in use, then the specific binding is removed.
2) The Dock may implement an idle timer, deleting the binding if no traffic has been detected for a specified amount of time. This time is specified in the Visa.
3) When the Tether is terminated, for any reason, all bindings which use that Tether's address are removed.
4) When the Docking Session is terminated, or the Dock detects that the Adapter is no longer in service, then the Dock removes all bindings that use Tethers to that Adapter.

The distribution of bindings to other elements of the ZPR network is the responsibility of the Admin Service.

3.22. ZPR Header Sequence Number Management

All ZPR packets carry sequence numbers for replay protection.

Sequence numbers are per Security Association. They MUST be long enough so that they cannot roll over within the lifetime of the security association (we recommend that they be 64-bits long as a counter of that size will not roll over for any realistic packet rate). If a sequence number gets close to wrapping, a new security association (with new keys and, by definition, a new sequence number that starts at 0) MUST be installed.

Not all bits of the sequence number may be carried in the ZDP header. The entire sequence number, even if only some bits are included in the header, are included in the MICV.

The sequence number is an unsigned counter that increases by 1 for each packet sent.

The sequence number is initialized to 0 when the security association is created.

When a packet is transmitted, the transmitter increments the sequence number. If it wraps, the transmitter MUST A) drop the packet, B) tear down the current SA (sending no more packets via the SA) and C) establish a new SA (with new keys and a reset sequence number). If the Sequence number does not wrap, then the transmitter inserts the low-order n (n is specified in the ZDP configuration) bits into the ZDP header. When the MICV is calculated, the high-order bits are logically prefixed to the ZDP header and included in the MICV.

The receiver MUST ensure that the sequence number of a received packet is not the same as the sequence number of any other packet received on the SA. How this is accomplished is an implementation matter, but typically it would be done with a sliding-window scheme as described in the ESP RFC -- RFC4303[9].

3.23. Link or Docking Session Failure

If a Link or Docking Session is deemed to have "failed" then the following MUST occur:

1) If the Admin Service can be reached, then it MUST be notified of the failure. This notification should include as much diagnostic information as possible, such as the reason for declaring the failure, any statistical counters that lead to the decision, and so on. If the Admin Service cannot be reached, then this information SHOULD be stored locally and transmitted to the Admin Service when it can be reached.
2) Any Forwarding Table or Dock Forwarding Table entries with PEPs that direct traffic to the failed Link or Docking Session MUST be deleted.
3) The Node MUST attempt to restore the Link or Docking Session.

3.24. Crypto/Security Procedures

3.24.1. ZPR Parameters and the ZPI

ZPR make uses of security associations (SAs) to manage cryptographic keys, algorithms, and other parameters of the cryptographic algorithms. Among other things, the SA specifies:

- A session key to use to encrypt/decrypt the ZDP header,
- A session key to use in generating and checking the message integrity check value (MICV) of the ZDP header,
- The cryptographic algorithm to use,
- The MICV algorithm to use,

[9] Kent (2005)

- The size of the MICV value generated by the MICV algorithm,
- Which bits of the MICV value are actually placed in the header,
- The block size of the cryptographic algorithm.

SAs are generated by the ZPR key management protocols[10] or set directly by the ZPR Admin Service.

SAs have two conceptual states:

- Active – the SA may be used to secure and transmit a packet. Multiple SAs may be active at the same time. If so, the transmitter SHOULD randomly select an SA to use with any given packet.
- Inactive – the SA may not be used to transmit a packet. The receiver, however, MUST be prepared to receive and process a packet that specifies an inactive SA. This allows packets using a to-be-deleted SA to drain from the network and be successfully processed prior to removing the SA.

The SA controlling the cryptography applied to a particular ZDP packet is indicated by the ZPI (ZDP Parameters Index) field of the ZDP header (§ 4.2.4.1). Multiple ZPIs may specify the same SA.

The ZPI's scope is the individual Link or Docking Session.

A given ZPI value MUST NOT be changed to refer to a second SA until packets using the first SA can be presumed to have drained from the network. If the Link operates directly over a data link (that is, does not run over IP) then the packets will drain fairly quickly – on the order of 100 or 200 milliseconds. If the Link runs over IP, it is conceivable that packets may take several seconds.

3.24.2. ZPI 0

ZPI 0 is used by two components of ZDP:

1) First, the key management protocol. The protocol self-secures so there is no need for the security provided by an SA. In addition, the protocol does an initial key exchange when bringing up a Link or Docking Session and no other SA is available for securing the messages.
2) ZPR ARP uses ZPI 0 since it must execute before the keying protocol executes. Thus, there can be no security associations available. ZPR ARP's resolutions are not guaranteed to be secure. Since the keying protocol does the initial key exchange immediately after ZPR ARP resolves the addresses, the key exchange verifies the identity of each peer to the other.

This ZPI is used for the key management protocol's messages and ZPR ARP messages. It MUST NOT be used for any other messages.

ZPI 0 is a NULL SA, with no security. It is defined as having:

- Null encryption,

[10]TBD

- The MAC is an unkeyed Internet standard checksum over the protected bytes (providing basic error detection),
- The MICV is 2 bytes long,

The rationale for this is that A) when bringing up the first SA there are no other SA's available to use and B) Key management protocols are self-securing.

3.25. Endpoint Packet Compression and Expansion

When Endpoint Packets are carried through the ZPR Network they are compressed. This is done:

1) To reduce the size of the Endpoint Packet which reduces the overhead of encapsulating the Endpoint Packet in a ZDP Transit Packet,
2) To remove (some) predictable invariant information from the Endpoint Packet increasing the packet's entropy and thereby the strength of the encryption.

Currently only TCP/IP or UDP/IP packets are compressed.

Compression is done in two parts:

First, the Ingress Adapter ALWAYS removes the IP Addresses from packets it receives from the Endpoint. These addresses are bound with the Ingress Dock via the Bind Endpoint Addresses message (§ 4.3.9). This informs the Dock that packets for Stream ID *S* have source and destination IP addresses *SA* and *DA*. These addresses are passed to the Egress Dock, which then notifies the Egress Adapter of the binding. The Egress Adapter then reconstitutes the original Endpoint Packet by putting the addresses back in the packet.

Compression lessens the impact of ZDP headers. For IPv4 packets, the 8 bytes of address represents a significant part of the ZDP header (8 out of 9 bytes), not counting the MICV. For IPV6, the entire ZDP header is provided for, as well as about 1/2 of a typical MICV value. This reduces the likelihood of the packet from growing too large and causing an MTU problem someplace in the network.

Second, additional fields MAY be compressed out of the header. This compression is performed by the Ingress Adapter (and decompression by the Egress Adapter). The Visa directs which additional fields are compressed out.

Only fields that do not change may be compressed out.

3.25.1. IPv4 Header

Version Is 4 bits long and a constant, 4 (or 6 for IPv6). This field must be known to the Visa Service and may be compressed out of the Endpoint Packet.

Header Length This field is four bits long and indicates the size of the IPv4 header. If there are no options, it can be compressed out, otherwise it must be carried in the Transit Packet.

TOS This field is not predictable and must be carried in the CAP (Compressed Endpoint Packet).

Total Length Can be recalculated by the Egress Dock after expanding the CAP.

Identification This field must be carried by the CAP as it is not necessarily predictable.

Flags Contains a 0-bit and two bits indicating the status of IP fragmentation. There are two common values, 000 and 010, indicating that the packet is not a fragment, with the don't fragment bit either set or clear. This field could be compressed.
Fragment Offset This field is 0 for the first fragment of a packet or when the packet is not fragmented (the common case). Thus, it could be reduced to a single bit indicating whether it's 0 or not. If not, the full value must be carried.
TTL This field cannot be predicted and must be carried.
Protocol This must be known to the Visa Service in order to grant a Visa and can be sent to the Egress Dock as part of the Visa Distribution.
Header Checksum This can be recalculated after the IPv4 header is decompressed.
Source IP Address Is always compressed out by the Ingress Adapter and restored by the Egress Adapter.
Destination IP Address Is always compressed out by the Ingress Adapter and restored by the Egress Adapter.
IP Options Are generally not predictable/knowable and must be sent in the CAP.

The compressed IPv4 header format is

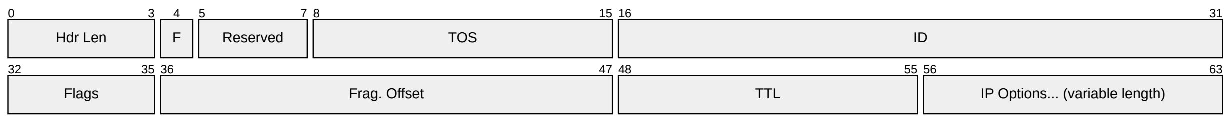


Figure 3.5: Compressed IPv4 header

Where:

HHHH Is the IP header length.
F Fragmentation Information Present bit. If set(1) the Flags and Frag. Offset fields are present, if clear(0) they are not.
TOS Is the TOS byte from the original IP header.
ID The ID field of the original IP header.
TTL The TTL of the original IP header.
Flags The flag bits from the original IP header. These bits are present only if the F bit is set.
Frag Offset The fragment offset field of the original IP header. This field is present only if the F bit (above) is set.
IP Options The IP options from the original Endpoint Packet. This field is present if and only if the HHHH field value is > 5. The length of the options is calculable from the IP Header length – (HHHH-5)*4.

3.25.2. IPv6 Header

Version Is a constant 6 and, thus, can be set by the Egress Dock.
Traffic Class Are generally not predictable/knowable and must be sent in the CAP.
Flow Label Are generally not predictable/knowable and must be sent in the CAP.
Payload Length Can be recalculated after expanding the CAP.
Next Header Must be known to the Visa Service in order to grant the Visa and can be sent to the Egress Dock as part of Visa Distribution.
Hop Limit Are generally not predictable/knowable and must be sent in the CAP.

Source IP Address Is always compressed out by the Ingress Adapter and restored by the Egress Adapter.

Destination IP Address Is always compressed out by the Ingress Adapter and restored by the Egress Adapter.

The compressed IPv6 header format is

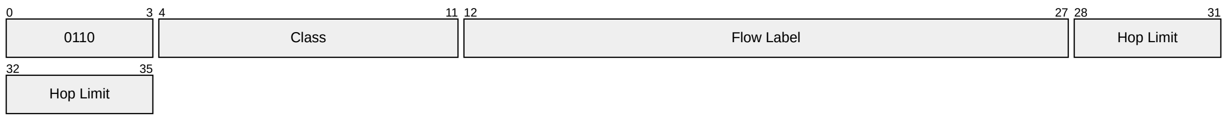


Figure 3.6: Compressed IPv6 header

Where:

0 1 1 0 Is the IP version number. Not strictly needed, but since the header must be an integral number of bytes long, cannot be removed.

Class IPv6 Traffic Class.

Flow Label The IPv6 Flow Label from the original Endpoint Packet.

Hop Limit The IPv6 Hop Limit from the original Endpoint Packet.

3.25.3. UDP Header

The UDP header consists of the following four fields:

Source Port This field is 16 bits long. It is known by the Visa Service and is sent to the Egress Dock as part of Visa Distribution.

Destination Port This field is 16 bits long. It is known by the Visa Service and is sent to the Egress Dock as part of Visa Distribution.

Length Is 16 bits long and is the length of the UDP header and its data. This can be recalculated based on the Payload Length field of the ZDP packet.

Checksum This field is 16 bits long. UDP specifies that packets may be sent without a checksum (by setting the checksum field to 0) and a single bit can be used to signal whether the checksum is 0 or not. Doing this compression is not seen as crucial since A) UDP traffic is a small portion of overall traffic and B) sending UDP traffic with a 0 checksum is generally discouraged, so the amount of traffic that can benefit from this compression would be small. As a result, the UDP checksum will always be sent.

The compressed UDP header format is:

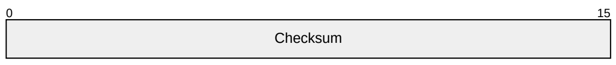


Figure 3.7: Compressed Udp header

3.25.4. TCP header fields

Source Port This field is 16 bits long. It is known by the Visa Service and is sent to the Egress Dock as part of Visa Distribution.

Destination Port This field is 16 bits long. It is known by the Visa Service and is sent to the Egress Dock as part of Visa Distribution.
Sequence Number This field cannot be compressed out since packets could be lost in the network.
Ack Number This field cannot be compressed out.
Data Offset Indicates the length of the TCP header. This can be compressed out if the TCP header does not include TCP options (the offset is then 5). If there are options, the full value must be carried.
Flags The flag bits (and associated MBZ bits) are not compressible.
Window Size This field is 16 bits long and cannot be compressed out.
Checksum The checksum is 16 bits long and can be recalculated by the Egress Dock after decompressing the Compressed Endpoint Packet.
Urgent Pointer Cannot be compressed out.
TCP Options Options are probably not worth compressing out.

The compressed TCP header format is

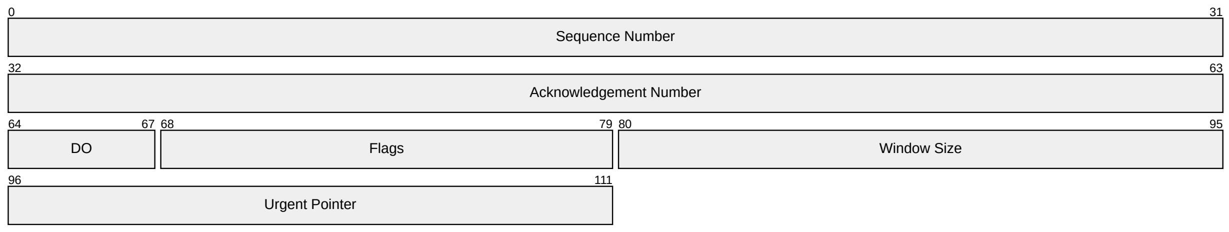


Figure 3.8: Compressed TCP header

Where:

Sequence Number and ACK Number Are the TCP Sequence and ACK numbers.
DO Are the data offset bits.
Flags Are the TCP Flag bits.
Window Size Is the window size from the original Endpoint Packet's TCP Header.
Urgent Pointer The Urgent Pointer from the original TCP packet.

3.26. ZPR Authentication

ZPR Authentication is the process by which an Endpoint is authenticated to the ZPR network. It should not be confused with the cryptographic authentication that is done by Nodes and Adapters as a part of instantiating the security associations necessary to bring up a Docking Session or Link. ZPR Authentication occurs after the Nodes have established their Links and the Agent and Dock have established their Docking Session.

After the Docking Session is established, a security association exists between the Dock and Adapter. The Dock and Adapter have verified each other's identity using cryptographically strong methods (such as validating X.509 certificates) and session keys have been generated.

There are two ZPR Authentication procedures:

1) A client Endpoint is connecting to the ZPR Network and attempts to establish commu-

nications with a server on the ZPR Network (*e.g.*, attempting to connect to an internal corporate web server or email server), and

2) A server[11] Endpoint that is on the ZPR network is starting and wishes to accept incoming requests from client Endpoint on the ZPR network.

The sequence of operations is:

1. **Docking Session Authentication**

   The Adapter and dock authenticate to each other and establish session keys for later ZDP messages. Any scheme can be used so long as the results are that the Adapter and Dock know each other's identity[12] with cryptographic certainty and have established session keys to use in encrypting ZDP traffic.

2. **Hello Request**

   The Adapter sends a Hello Request (see § 4.3.4) to the Dock. The information included in the Hello Response is set by policy. It may include things like a protocol version, hardware asset number, GPS location, and so on.

   a. **Evaluate Hello Request** The Dock evaluates the Hello Request to decide whether to accept the request or not. This evaluation MAY take into account current conditions on the Dock (*e.g.*, its current load) and MAY involve other entities in the network.

      If the Dock elects not to proceed then the process aborts and a Hello Response indicating failure is sent to the Adapter. The Docking Session is NOT terminated.

3. **Authentication Visa Request**

   The Dock requests that the Visa Service install a constrained Visa that will allow the Adapter to send ZDP messages to, and receive them from, the Auth service. These messages are used to establish strong ZPR authentication.

   This request includes identity information established in step 0 as well as any relevant information in the Hello Request.

   The Visas are constrained because the Endpoint has not been strongly authenticated. The constraints limit the data rate and message types in order to mitigate attacks.

   The Visa service may decline the request in which case this process ends. The Docking Session is NOT terminated.

4. **Hello Response**

   The Dock Sends a Hello Response to the Adapter indicating success. The response must include:

   - An Endpoint Authentication ZPR Address (EAZA): This is the ZPR Address that the Adapter uses as the "source" address for all packets sent to the Auth Service.

[11]The terms "client" and "server" are used in the classic networking sense.

[12]This is a "low-level" identity, not the ZPR/ZPL Identity. Its purpose is solely to ensure that the Adapter and Dock "know" what is at the other end of the Docking Session.

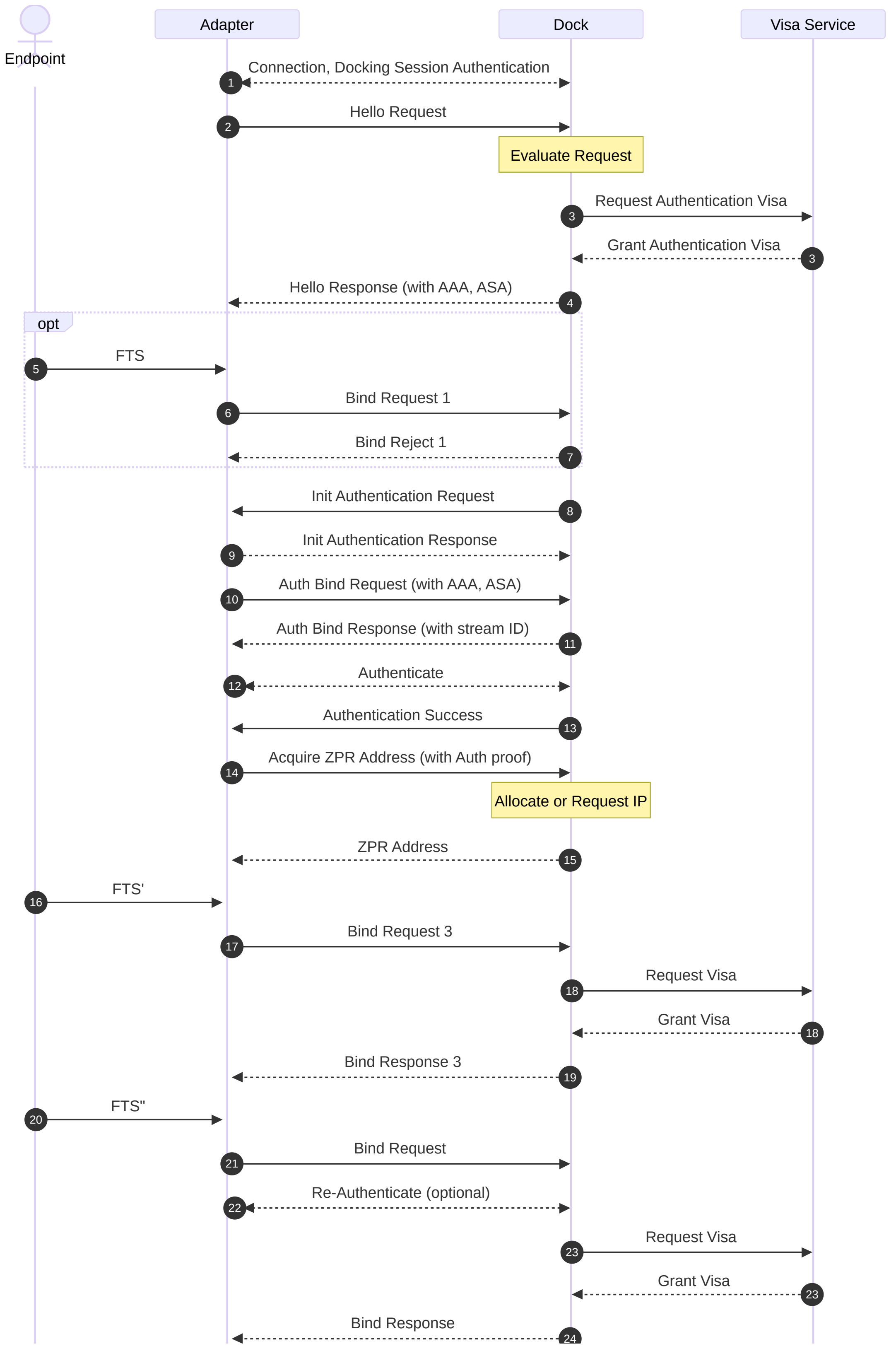

Endpoint
Adapter
Dock
Visa Service
1 Connection, Docking Session Authentication
2 Hello Request
Evaluate Request
3 Request Authentication Visa
3 Grant Authentication Visa
4 Hello Response (with AAA, ASA)
opt
5 FTS
6 Bind Request 1
7 Bind Reject 1
8 Init Authentication Request
9 Init Authentication Response
10 Auth Bind Request (with AAA, ASA)
11 Auth Bind Response (with stream ID)
12 Authenticate
13 Authentication Success
14 Acquire ZPR Address (with Auth proof)
Allocate or Request IP
15 ZPR Address
16 FTS'
17 Bind Request 3
18 Request Visa
18 Grant Visa
19 Bind Response 3
20 FTS''
21 Bind Request
22 Re-Authenticate (optional)
23 Request Visa
23 Grant Visa
24 Bind Response

Figure 3.9: Authentication Packet Flow

- An Authentication Service ZPR Address (ASZA): The ZPR Address that the Adapter MUST use as the “destination” address when sending ZPR Authentication messages to the Auth Service.

Other TLVs MAY be included in the Hello Response.

After the hello process, what happens next depends on what the Adapter, Endpoint and Dock do.

- The Endpoint may try to send a data packet to some other node using ZDP to start a connection or session of some sort. (We refer to this as the “First TCP Syn” – FTS – though it may not be, in fact, a TCP Syn),
- The Dock may decide, for whatever reason, to initiate authentication with the Adapter/Endpoint. If the Endpoint is an application server and does not send FTS’s then this method is used to get the Endpoint/Adapter up and running, or
- The Adapter may wish, for some reason other than the above, to authenticate. The Adapter can use this to get an application server Agent up and running.

---

5. **Endpoint sends FTS**

   The application sends the FTS packet. The packet is determined to be an FTS when the Adapter looks up the packet in its forwarding tables in order to determine the PEP and Stream ID to use in forwarding the packet, and it does not find an entry for the packet.

6. **Bind Request 1**

   The adapter sends a Bind Endpoint Address (see § 4.3.9) to the Dock with the FTS packet. The packet is not buffered, it is discarded.

7. **Bind Reject 1**

   The Dock notes that the Adapter/Endpoint has not performed the required ZPR Authentication. It rejects the Bind Request with a “Not Authenticated” error code.

8. **Init Authentication Reqquest**

   The Dock sends an Init Authentication (§ 4.3.15) message to the Adapter. The Bind Reject causes the Dock to send this. The Dock may send this message for other reasons as well (in particular, it may do this so that a server can authenticate itself to the ZPR Network.

   The purpose of this message is to get the Adaptor/Endpoint to authenticate itself with the ZPR Auth Service.

9. **Init Authentication Response**

   Acknowledges receipt of the Init Authentcation Request.

10. **Authentication Bind Request**

    The Adapter sends a second Bind Request to the Dock (see § 4.3.9). This request includes just the Endpoint Authentication ZPR Address as the source address and Authentication Service ZPR Address as the destination.

The intent here is to get a Stream ID for the Adapter to use in communicating with the Auth. Service.

11. **Authentication Bind Response**

    The Dock evaluates the Bind request. Since the request matches the two ZPR addresses associated with the constrained Authentication Visa (see step 3) the Dock installs a PEP, etc. and returns a Stream ID for the Adapter to use when communicating with the Auth. Service in the following steps.

12. **ZPR Authentication**

    The Adapter performs authentication with the Auth. Service. This is a series of authentication messages that are exchanged by the Adapter and the Auth Service. These messages travel over the ZPR Net as ZDP packets, using the Stream ID obtained in the "Bind Response 2" step, above. These packets use the Visa installed in step 3.

    These messages are normal ZPR Data Protocol messages.

    The authentication protocol is presumed to end-to-end encrypt, or otherwise protect, its messages so that they are not visible to attackers, spoofable, etc.

    Note Also that the Adapter/Endpoint may decide, for any reason of its own, to initiate authentication and start this process. (As described above, the Adapter, knowing its Agent is an application server, may start the process so that the server is connected to the network).

    If this authentication process starts without having received an FTS, then it is a 'generic' authentication. Presumably it might be good for many different connections, none of which are specified.

13. **Authentication Finishes**

    Eventually the Authentication process ends with a "Successfully Authenticated" message to the Adapter. This message includes Authentication Token (the Blob). (This message is actually a part of the exchange of messages in step 14 but is called out separately to indicate that it includes the blob).

14. **Acquire ZPR Address Request**

    The Adapter requests the ZPR Address for the identity that was proven in the Authentication Steps 14 & 15, above. It includes the Blob it received in step 15 as proof that it is the indicated identity. See § 4.3.16.

    a. **ZPR Address Allocation**

       Based on configuration and policy, the Dock will either be allowed to allocate an address from its own pool or need to reach out to another service to acquire an address for this endpoint.

15. **Grant ZPR Address**

    The Dock sends the ZPR Address to the Endpoint/Adapter. See § 4.3.17 via a Grant ZPR Address message.

16. **FTS Retransmission**

If the process started by the Adapter receiving the First TCP Syn (FTS) from the Endpoint, the Endpoint will eventually retransmit the SYN (FTS'). The Adapter receives the message and attempts to find a Stream ID to use for it.

17. **Bind Request 3**

    The Adapter does not find a Stream ID to use, so it sends a Bind Request (see § 4.3.9) to the Dock.

18. **Visa Install**

    The Dock requests a Visa from the Visa Service. The Visa Service creates a Visa and installs it in the ZPR Network. The authentication needed to allocate this Visa was done in step 14.

19. **Bind Response 3**

    When the dock has been notified that the Visa has been installed it sends a Bind Response (§ 4.3.9) to the Adapter, giving it the Stream ID to use.

    Data may then flow through the network.

20. **FTS"**

    If a different FTS is sent by the Endpoint, it will require a different authentication, etc. round. Since the constrained visa is in place, the steps required to install it are not performed.

3.27. Path MTU Management

Whenever a Node (Egress Adapter, Egress Dock, Forwarder, or Ingress Dock) learns of a downstream MTU for a specific Stream ID it propagates that MTU upstream. This process repeats until the Ingress Adapter learns the MTU. The Ingress Adapter then can either A) inform the Endpoint of the MTU via an appropriate ICMP message, if and when the Endpoint sends an Endpoint Packet that exceeds the MTU or B) perform IP fragmentation on behalf of the Endpoint.

Each Set Path MTU Request sent to an upstream is acknowledged with a Set Path MTU Response sent back downstream. The messages follow the Request/Response semantics described in § 4.4.

If the upstream must use an MTU that is smaller than the requested MTU for some reason it indicates the MTU it will use in the response message. The downstream ignores this, but MAY use it in a log message.

If the upstream must use an MTU that is *larger* than the requested MTU for some reason then it MUST A) report this to the admin servers B) indicate it to the downstream via the response message, and C) Terminate the stream[13].

[13]It appears that this would be a deadlock situation. Alternate strategies are sought!

3.28. Error Reporting

Many ZDP operations can fail. If they do the question of how to report the failure arises, in particular "should the error be reported to the peer?". This question is important because the failures could be because the peer is an attacker. Attempting operations and interpreting the response codes *may* give the attacker information to help mount a successful attack.

The protocols include error messages which contain fields that can describe errors and problems in detail. An implementation MAY elect to keep this information to a minimum, perhaps even reporting all errors as "other". An implementation MAY also elect to always report "success" to the peer, while in fact "spoofing" the peer from that point onward (discarding any received transit packets and always reporting "success" for any future management operation received from the peer).

Policies control how errors are reported to peers.

Errors MUST always be logged (as directed by Logging Policies).

3.29. Ingress Adapter Classification

When an Ingress Adapter receives an Endpoint Packet from its Endpoint, it classifies to the packet to A) Determine whether a Tether exists for that packet or not and, if so, B) Which Tether to assign the packet to (that is, what Stream ID to assign to the packet).

The classification rule to use is installed via the Bind Endpoint Address Response message (§ 4.3.9). The message allows for different types of classification specification. Currently, only type 0 is supported. This type is a simple "compare for equal" against a specified 3-, 4-, or 5-tuple.

The type-0 classification specification includes the following fields:

4 Whether the packet is IPv4 or IPv6 (and, therefore, whether the Endpoint Addresses are 32 or 128 bits long),

S Whether the source port is to be included in the classification or not,

D Whether the destination port is to be included in the classification or not,

IP Protocol The IP protocol the packet must have to be accepted by the classifier,

Source and Destination Endpoint Address The source and destination Endpoint Addresses the packet must have in order to be accepted by the classifier,

Source Port The source UDP or TCP port that the packet must have in order to be accepted by the classifier, and

Destination Port The source UDP or TCP port that the packet must have in order to be accepted by the classifier.

Conceptually the classification algorithm is:

```
for each rule {
    if packet.ipversion != rule.V {
        next rule
    }
    if packet.source_address != rule.source_endpoint_address {
        next rule
    }
```

```
        if packet.destination_address != rule.destination_endpoint_address {
            next rule
        }
        if (packet.ip_protocol != rule.ip_protocol) {
            next rule
        }
        if (packet.ip_protocol == TCP or UDP) {
            if (rule.d == 1 && packet.destination_port
                               != rule.destination_port)
            {
                next rule
            }
            if (rule.s == 1 && packet.source_port != rule.source_port) {
                next rule
            }
        }
        this rule accepts the packet, handle per the PEP associated with
        the rule
    }
    This packet is not accepted by any rule, assume it is the first packet
    for a new flow
```

Note that the algorithm is presented as a linear search through the classification table. We presume that an implementor will use a more efficient method, such as a tree search.

The type-0 classifier does not allow for multiple values for any field (eg. a range of TCP ports or an IP address prefix). This is left for later classifier types. Allowing multiple values can improve performance in that a Dock or even Adapter, could admit a new flow without having to go through ZPR Authentication, Visa distribution, and so on.

4. ZPR Protocol Packet Formats

This chapter defines the packet formats for ZDP.

4.1. General Considerations

Packets are always “packed”, even though a packet format diagram may make it appear otherwise. For example, if a packet format is diagrammed as:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+
| One-byte-field|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Four byte field                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The packet is sent as five bytes:

```
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| One-byte-field| Four byte field                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

In packet diagrams, the start of the packet is the topmost/leftmost component. The transmit order of the above 5-byte packet is as shown in the above diagram.

Packets are diagrammed and sent in “big endian” order (aka network byte order). For example, a four-byte field containing the numeric value 1234510 (0x3039, or 0b00110000 00111001) would be:

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 1 1 1 0 0 1|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

4.2. ZPR Data Protocol (ZDP)

ZDP is a common protocol used to carry all traffic within ZPR: application data in Endpoint Packets (both between Dock and Adapter as well as Forwarder-to-Forwarder) and ZPR Management Messages.

4.2.1. ZDP Classes

ZDP defines the following packet classes:

Transit Packet on a Link or Docking Session This packet carries an Endpoint Packet from an Ingress Adapter to an Egress Adapter via a sequence of Tethers, Docks, Forwarders and Links. The Endpoint Packet itself is compressed (§ 3.25) and cryptographically protected via the A2A Security Association (§ 4.2.2). The ZDP Header is cryptographically protected by the ZDP Header Security Association (§ 4.2.2).

Per Flow Management Packets on a Link or Docking Session These are management packets that traverse Links or Docking Sessions and are used by the adjacent Forwarders or Dock and Adapter to manage a Flow identified in the packet. The packets involved in Visa Heralding (§ 3.11 and § 4.3.5) are examples. These packets are cryptographically protected via the ZDP Header SA (§ 4.2.2).

Non-Flow Management Packets on a Link or Docking Session Per-Link or per-Docking Session management packets that do not apply to a specific stream. These packets go between adjacent Forwarders or between a Dock and an Adapter. They are cryptographically protected via the ZDP Header SA (§ 4.2.2).

Per-flow and non-per-flow management packets are disambiguated by the type field's value in the header.

4.2.2. ZDP Security Associations

ZDP makes use of two different classes of security association (SA):

ZDP Header Security Association (ZHSA) This SA exists between an Adapter and the Dock with which it has a Docking Session or between adjacent Forwarders over a Link. The ZHSA protects the ZDP header of all packets traversing the Docking Session or Link. For management packets it also covers the management message. The ZHSA DOES NOT cover encapsulated Endpoint Packets. The SA provides data integrity, origin authenticity, data confidentiality, and replay protection for all packets containing management messages.

The SA is managed on a pair-wise basis between the Adapter and Dock or adjacent Forwarders. Management can be via a specific keying protocol (currently specified as IKEv2) or the Visa distribution service. The SA is created as a part of the process to bring up a Link or Docking Session.

The ZHSA indicates how much of the ZDP packet is encrypted by the ZHSA. There are two options: Partial and Full. In Partial encryption, only the Type, Excess Length, Sequence Number, Stream ID, and padding fields are encrypted. In Full encryption the entire ZDP packet, except for the ZPI byte and MICV fields is encrypted.

This flag is required because the amount of the packet that is encrypted depends on the packet type. Transit Packets have only the ZDP header encrypted and covered by a MICV. All other packets are completely encrypted and covered by the MICV. The issue is that the type field is encrypted so it cannot be used to determine whether how much of the packet is protected – thus, the information is stored in the SA.

The ZHSA field in the ZDP header selects the header security association to use when processing the packet.

ZHSAs are created and managed either A) by trusted admin services manually configuring the ZHSAs in the Nodes or B) by a ZDP keying protocol (TBD).

Note Well: The ZHSA is NOT ZPR Authentication.

Between Ingress and Egress Adapter (A2A SA)[1] This SA exists between the Ingress and Egress Adapters of a given flow of Endpoint Packets. It provides cryptographic protection of the Endpoint Packet as it traverses the network. It allows the Egress Adapter to verify that the Endpoint Packet arrived at the Egress via the Visa by which the packet was admitted at the Ingress Adapter. The A2A SAs do not provide neither replay protection. They MAY provide confidentiality.

The A2A SA is managed by the Visa Service. When a Visa is granted, the VS explicitly generates the necessary cryptographic material and directly sends this material to the Ingress and Egress Docks. The Docks, in turn, send the keying material to their respective Adapters via the Bind Endpoint Response message (§ 4.3.9) It is not propagated from Ingress Dock to Egress Dock via the Visa Heralding procedure; thus, the material is known only by the Visa Service and two Adapters.

The keying material MUST be encrypted by the Visa Service so that only the Adapters can access it.

This combination of SAs

1) Cryptographically protects those parts of ZDP packets that are used by the ZPR network, thus protecting the ZPR network itself and
2) Minimizes the resources required for encryption, decryption, and integrity protection by protecting Endpoint Packets on an Adapter-to-Adapter basis and not Forwarder-to-Forwarder.

4.2.2.1. The NULL Security Association

There also is a NULL Security Association that is available for limited use when no other SAs are available. The NULL SA defines no encryption or authentication. Packets are transmitted in the clear and are presumed to be authentic and unmodified. The only valid uses of the NULL SA are

1) In ZDP ARP, where it is used when doing a ZPR ARP exchange before strong SAs have been brought up. This is not a significant security hole since immediately after doing the ARP exchange, the two nodes bring up a Link or Docking Session, which includes cryptographically authenticating each other and creating strong SAs. This will detect any spoofing done via ZPR ARP.
2) For the ZDP keying protocol. This is not an issue because the key exchange must operate before any SAs are established and the protocol is inherently secure.

The NULL SA is selected by ZPI 0.

See also § 3.24.2.

The NULL SA may also be used when the substrate is a VPN and the VPN provides adequate security.

4.2.3. ZPR Versions

The Version Number field of the ZDP header indicates the ZDP protocol version. This document describes version 1 of ZDP.

Version 0 is reserved for experimental use.

Version 7 is reserved for future use.

The ZDP Version indicates the structure of the ZDP packets, including the lengths of many of the fields.

ZDP versions are defined by ZDP protocol specifications (such as this document).

NOTE WELL: the concept of the Configuration no longer applies to ZDP packet structure.

4.2.4. Base ZDP Packet Structure

4.2.4.1. Fields

All ZPR packets have the following header. It provides the information necessary to receive and disambiguate the packet. The header contains the following fields, in the given order:

Version A three-bit field that indicates the ZDP Version of the packet. This document describes version 1.

ZHSAID ZDP Header Security Association ID - identifies the security association used to process the ZDP header.

Type The type of message carried in the ZDP packet.

Excess Length A 1-byte field that is the difference between the size of the ZDP packet and the size of the Substrate Packet that contains the ZDP packet.

The field is an unsigned integer.

The motivation is that some Substrate Networks will have minimum packet payload sizes that may be larger than a ZDP message. For example, an Ethernet frame's data must be at least 46 bytes long) which might be longer than the encapsulated ZDP packet. This field represents the difference between the two. If a management packet (such as an echo message) is small, say 30 bytes, and is carried on an Ethernet, this field would contain 16.

```
ZDP packets MUST be transmitted in the smallest substrate packet possible, consistent with the S
```

Sequence Number This is a sequence number, used in doing replay protection.

> TODO: Change this for new reliability scheme.
>
> Sequence numbers are local to a ZDP Header SA.
>
> This field is 2 bytes long. The factor that controls the size of the field is how large a window is provided to accommodate legitimate packet misorderings. The available window for an n-bit field is 2^{n-1} packets, though 2^{n-2} is better. This means that the maximum size of the window is 32,768 (or 16,384) packets. This is the *maximum* size that the window may be; policies can set it smaller.

Stream ID This field is present for ZDP Transit Packets and stream-oriented management packets. Its exact use depends on the packet type (*e.g.*, for Transit Packets, the packet is switched based on the Stream ID).

The size of the Stream ID is declared in the Hello Response.

Management Data The presence of this field depends on whether the ZDP packet contains an Endpoint Packet message or is a Management Message, as indicated by the type field. If the packet is a Management Message, then this field contains data pertinent to the management operation being performed. The size and format of this data depends on the management operation.

If the packet contains an Endpoint Packet, then this field is absent.

Pad Padding used to make the ZPR header an integral number of encryption cipher blocks in length.

When generating a packet, the pad MUST be set to all 0's to avoid leaking internal data. On reception the pad MUST be checked and, if not all 0, the packet silently discarded and the Docking Session or Link brought down. Furthermore, the upstream should be "ignored".[2]

Header MICV Message Integrity Check Value calculated over the entire ZPR Packet header. The size of the MICV is determined by the MICV algorithm selected by the security association.

4.2.4.2. Encryption and MICV Calculation

When transmitting a ZPR packet, the packet is encrypted and then the MICV is calculated. On reception the MICV is checked and, if the check fails, the packet is discarded. After checking the MICV, the packet is decrypted.

The MICV is calculated over all fields except the Header MICV field.

All fields except the Version, ZHSAID, and Header MICV fields are encrypted.

4.2.4.3. Packet Structure

The format of the packet is:

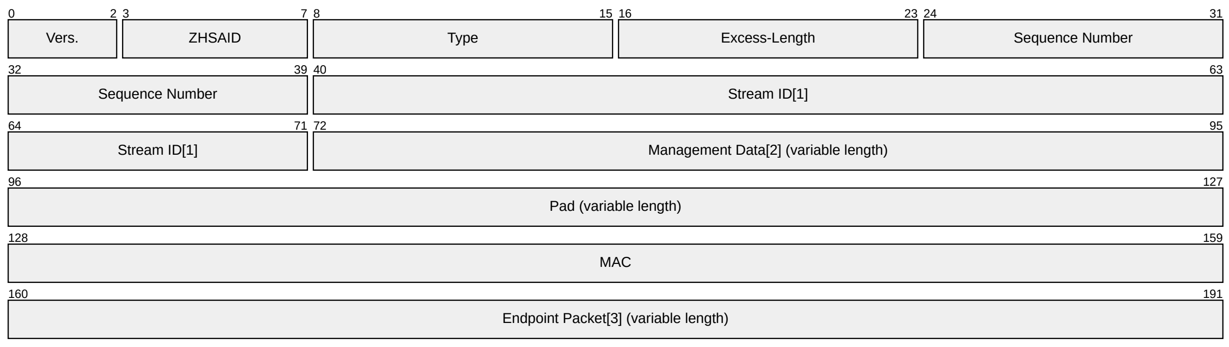


Figure 4.1: ZPR Packet Structure

Notes:

[1] Stream ID is present only if the packet contains an Endpoint Packet or contains a stream-oriented management message.

[2]This may seem excessive but if one of ZPR's motivating factors is security, having a field in a packet that has uncontrolled contents is an invitation to data exfiltration in some manner. Note that ZDP will operate properly if the contents of the pad are not controlled, so if this requirement is deemed onerous, it can be removed with no other deleterious effect.

[2] Management data is present only if the packet contains a management message.

[3] Endpoint Packet is present only if the packet is a Transit Packet.

4.2.4.4. Type Values and encodings

The following Type values are assigned.

Value dec./hex	Type	Flow-Oriented?	Ref.
0/0x00	Transit Packet	Yes	§ 4.2.5
1/0x01	Unused	Yes	
2/0x02	Destination Unreachable	Yes	§ 4.3.13
3/0x03	Visa Herald Request	Yes	§ 4.3.5
4/0x04	Visa Herald Response	Yes	§ 4.3.5
5/0x05	Unused		
6/0x06	Unused		
7/0x07	Visa Retract Request	Yes	
8/0x08	Visa Retract Response	Yes	
9/0x09	Stream ID Withdrawl Request	Yes	§ 4.3.7
10/0x0a	Stream ID Withdrawl Response	Yes	§ 4.3.7
11/0x0b	Bind Endpoint Address Request	Yes	§ 4.3.9
12/0x0c	Bind Endpoint Address Response	Yes	§ 4.3.9
13/0x0d	Unbind Endpoint Address Request	Yes	§ 4.3.9
14/0x0e	Unbind Endpoint Address Response	Yes	§ 4.3.9
15/0x0f	Authentication Request	Yes	
16/0x10	Authentication Response	Yes	
17/0x11	Set Path MTU Request	Yes	§ 4.3.12
17/0x11	Set Path MTU Response	Yes	§ 4.3.12
18/0x12… 95/0x5f	Unallocated	Yes	
96-126/ 0x60-0x7e	Reserved for private use and experimentation	Yes	§ 4.3.15
127/0x7f	Reserved, must not be used Messages receive with this value MUST be silently discarded.	Yes	
128/0x80	ZPR ARP	No	§ 8
129/0x81	Key Management (*e.g.* IKEv2)	No	§ 7.1
130/0x82	Discard	No	§ 4.3.1
131/0x83	Echo Request	No	§ 4.3.2
132/0x84	Echo Response	No	§ 4.3.2
133/0x85	Terminate Link or Docking Session Request	No	§ 4.3.3
134/0x86	Terminate Link or Docking Session Response	No	§ 4.3.3
135/0x87	Terminate Link or Docking Session Indication	No	§ 4.3.3
136/0x88	Hello Request	No	§ 4.3.4

Value dec./hex	Type	Flow-Oriented?	Ref.
137/0x89	Hello Response	No	§ 4.3.4
138/0x8a	Configuration Request	No	§ 4.3.14
139/0x8b	Configuration Response	No	§ 4.3.14
140/0x8c	Acquire ZPR Address Request	No	§ 4.3.16
141/0x8d	unused	No	
142/0x8e	Acquire ZPR Address Response	No	§ 4.3.16
143/0x8f	Unused	No	
144/0x90	Unused		
145/0x91	Report	No	§ 4.3.11
146/0x92	Init Authentication Request	No	§ 4.3.15
147/0x93	Init Authentication Response	No	§ 4.3.15
148/0x94	Grant ZPR Address Request	No	§ 4.3.17
149/0x95	Grant ZPR Address Response	No	§ 4.3.17
150/0x96… 224/0xdf	Unallocated	No	
0xe0-0xfe 224-254	Experimental and Private Use	No	§ 4.3.15
255 (0xff)	Reserved. Must not be used. Messages received with this value MUST be silently discarded.	N/A	N/A

Table 4.1: Stream ID Response Error Codes

ED NOTE: THIS TABLE MAY NOT REFLECT THE USAGE OF CODE POINTS IN THE EXPERIMENTAL IMPLEMENTATION. CODE POINT ASSIGNMENTS WILL BE FINALIZED PRIOR TO PUBLIC RELEASE.

By convention, type values with the high-order bit cleared (0x00-0x7f) all contain a stream ID immediately following the common header. Type values with the high order bit set (0x80-0xff) do not contain a stream ID.

By convention, request messages all have the low-order bit (0x01) set, responses (or messages for which there is no request/response method) have the low order bit cleared.

These values apply to all ZDP packets, whether being carried over a Link or a Docking Session.

4.2.5. Transit Packets

This section specifies the encapsulation of Compressed Endpoint Packets as they are transported across both Docking Sessions and Links in Transit Packets.

4.2.5.1. Fields

The Transit Packets are comprised of the following fields:

Version A three-bit field that indicates the ZDP Version of the packet. This document describes version 1.

ZHSAID ZDP Header Security Association ID - identifies the security association used to process the ZDP header.

Type Packet type. Is 0 for a Transit Packet. See § 4.2.4.1 and § 4.2.4.4.

Excess Length See § 4.2.4.1.

Stream ID See § 4.2.4.1.

Pad See § 4.2.4.1.

Header MAC See § 4.2.4.1.

A2A Security Association ID The Adapter-to-Adapter Security Association ID. This field selects the A2A SA. Since each Visa (and therefore each stream) has its own A2A SAs, the A2A SAID selects an SA within the context of the Visa by which the packet travelled.

The size of the A2A SAID field is determined by the MICV algorithm used.

This field is 1 byte long. This is adequate since A) We expect that there will be a relatively small number of A2A Security Associations in effect at any one time, and B) A2A SAID values can be reused.

See § 4.2.2

Compressed Endpoint Packet\ This field contains the Endpoint Packet in its entirety.

A2A MAC Message authentication code calculated over the entire Uncompressed Endpoint Packet. This MAC is calculated prior to compression (see § 3.25) by the Ingress Adapter and after decompression (see § 3.25) by the Egress Adapter.

The size of the A2A MAC field is determined by the MAC algorithm selected by the security association.

4.2.5.2. Packet Structure

The structure of the Transit Packet, is:

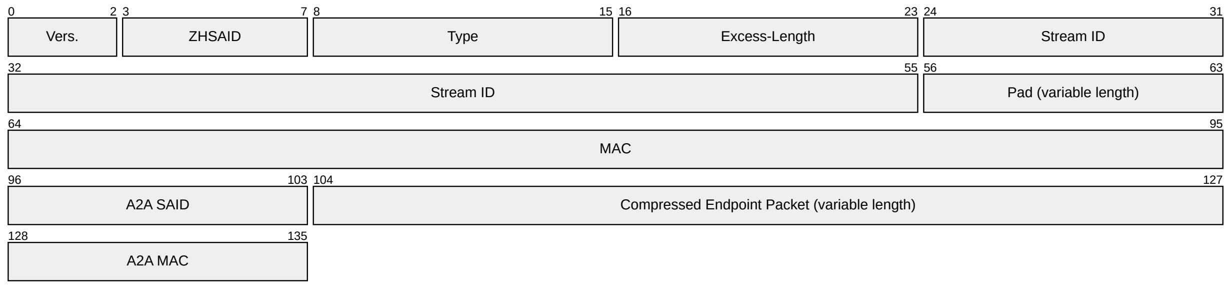


Figure 4.2: ZPR Transit Packet Format

4.2.6. Per-Flow Management Packet

Per Flow management packets contain a management message that pertains to a specific traffic flow.

The fields are described in § 4.2.4.1.

The format of the ZDP packet is:

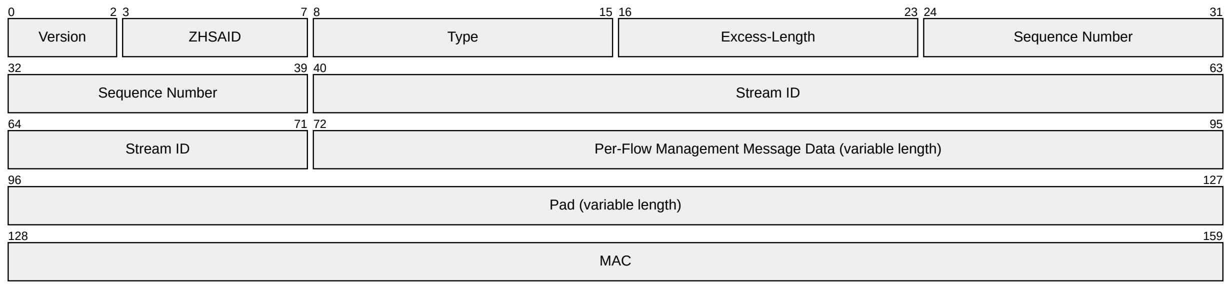


Figure 4.3: Per-Flow Management Packet Format

Note that

1) All messages with a stream ID in the header have a type field value on the range (0x00-0x7f, inclusive).
2) The length and structure of the Per-Flow Management Message Data field depend on the message. These messages, and their data, are specified in § 4.3.

4.2.7. Non-Per-Flow Management Packet

Non-Per-Flow management packets contain a management message that pertains to the entire Link or Docking Session or the associated Forwarders, Adapters, and Docks.

The fields are described in § 4.2.4.1.

The format of the Non-Per-Flow Management Packet is:

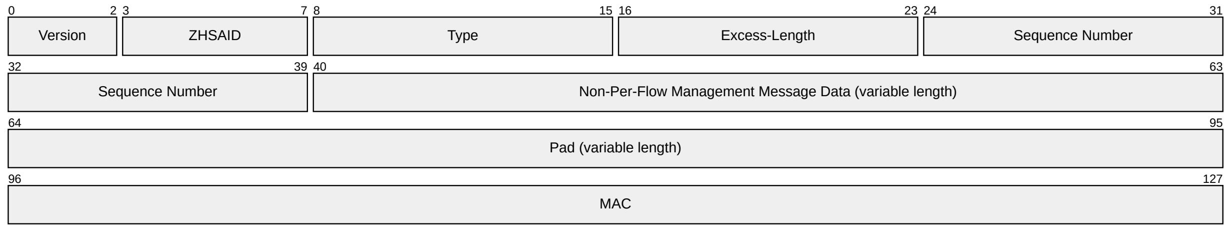


Figure 4.4: Non-Per-Flow Management Packet Format

Note that:

1) The value of the type field in all Non-Per-Flow management messages is in the range 0x80-0xff, inclusive.
2) The length and structure of the Non-Per-Flow Management Message Data field depends on the message. These messages, and their data, are specified in § 4.3.

4.3. Management Packets

This section specifies the management messages.

4.3.1. Discard Message

Discard messages are silently discarded. Typically, they are neither logged nor reported however an implementation MAY elect to log/report them, preferably at a “debug” level.

Discard messages are Non-per-Flow Management Messages and are carried in Non-Per-Flow Management Packets (§ 4.2.7).

There is no data to a management packet other than the ZDP header. In other words, the length of the Management Message field of the ZDP Packet is 0.

This message does not follow the request/response semantics outlined below.

The sequence number is provided to assist in determining packet loss and in reporting.

Discard messages are provided as an analogy to RFC863, “The Discard Protocol” for TCP and UDP.

4.3.2. Echo Request and Response

The Echo Request and Response Management messages provide a mechanism to send a message to a peer and receive a response without any side effects.

When a Management Echo Request is received, its type should be changed to Echo Response and then sent back to the sender of the request via the Link or Docking Session over which the Request was received.

Echo Request and Response messages are Non-Per-Flow Management Messages and are carried in Non-Per-Flow Management Packets (§ 4.2.7).

4.3.2.1. Echo Request

The Echo Request is transmitted in the Management Message Data field of the Non-Per-Flow Management packet. The Management Message Data has the following format:

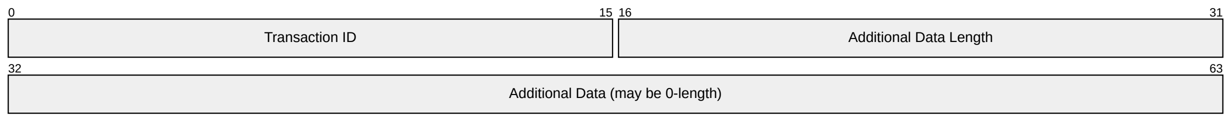


Figure 4.5: Echo Request Management Message Data Format

Where:

Transaction ID Is the Transaction ID number that uniquely identifies the Echo Request.
Additional Data Length Is the number of Additional Data bytes in the message. It may be 0.
Additional Data Is additional data that may be sent. The content is unspecified. If a request has additional data, then it must be returned in the response.

4.3.2.2. Echo Response

The Echo Response is transmitted in the Management Message Data field of the Non-per-Flow Management packet. The Management Message Data has the following format:

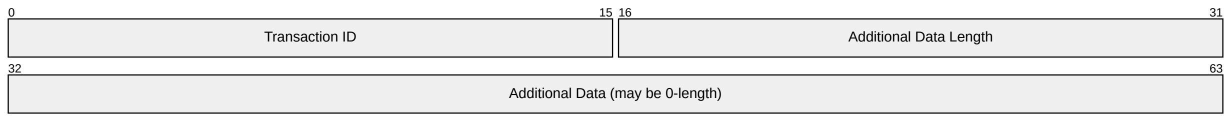


Figure 4.6: Echo Response Management Message Data Format

Where:

Transaction ID Is the Transaction ID of the Echo Request that this message is a response to.
Additional Data Length Is the number of Additional Data bytes in the message. It may be 0.
Additional Data Is additional data that was sent in the request.

Echo Request/Response messages may be used as keepalives or liveness detection (§ 3.17).

Echo request and response use the Request/Response semantics specified in § 4.4.

4.3.3. Terminate Indication, Request and Response

There are three Management messages involved in terminating a Link or Docking Session:

- Terminate Indication
- Terminate Request
- Terminate Response

The Indication and Request both inform the peer that the sender wishes to terminate the Link or Docking Session. The difference is that the Indication is imperative, informing the peer that the sender is unilaterally terminating the Link or Docking Session immediately and no response is expected. The Request informs the peer that the sender wishes to terminate but will wait until the peer returns a Terminate Response. (Of course, if the sender gets tired of waiting for a Response, it could send an Indication and terminate immediately).

The Indication and Request have the same format:

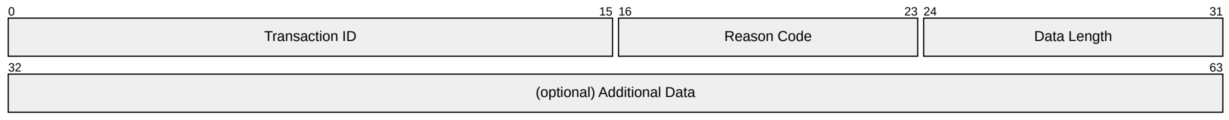


Figure 4.7: Terminate Indication and Request Management Message Data formats

Where:

Transaction ID Is a sequence number uniquely identifying the indication or request.
Reason Code A numeric reason code; one of the following:

Code	Name	Meaning
TBD	OTHER	This value is used when any of the following is inappropriate.
TBD	BAD_SEQUENCE_NUMBER	A sequence number in a response to an earlier request is not a sequence number to a valid request.
TBD	REQUEST_TIME_OUT	Some operation has timed out, resulting in termination of the connection.
TBD	RESET	A RESET event was issued against the Docking Session or Link (only occurs in indications).

Table 4.2: ZDP Termination Reason Codes

Data Length Is the number of Additional Data bytes in the message. It may be 0.
Additional Data Optional additional data (such as a human-readable message more fully explaining the termination).

The format of the Terminate Response Management Message Data is:

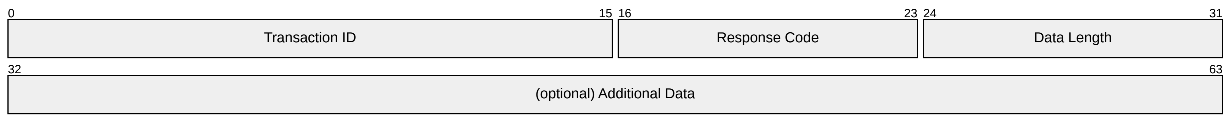


Figure 4.8: ZDP Terminate Response Management Message Data

Where:

Transaction ID Is a sequence number identifying the indication or request for which this is a response.
Response Code Indicates whether the termination will be done or not and, if not, why. The following response codes are defined:

Code	Meaning
0	Success – Link or Dock Session will be terminated.

1	OTHER. This value is used when any of the following is inappropriate or to avoid passing too much information to a peer.

Table 4.3: Terminate Response Codes

Data Length Is the number of Additional Data bytes in the message. It may be 0.
Additional Data Optional additional data (such as a human-readable message more fully explaining the termination response).

Terminate Request and Response messages use the Request/Response semantics specified in § 4.4.

4.3.4. Hello Request and Response

The Hello Message and response are sent once, when a Link or Docking Session comes up. Each Node sends a Hello Request to its peer and receives a Hello Response from its peer. These messages are used to verify the Link or Docking Session as a part of bringing it up.

The ZDP type field indicates that these are Hello Request and Response Messages.

The Hello Request and Response messages are Non-Per-Flow Management messages.

Link Hello Request and Response messages use the Request/Response semantics specified in § 4.4.

4.3.4.1. Hello Request Message

The Hello Request Message Data is:

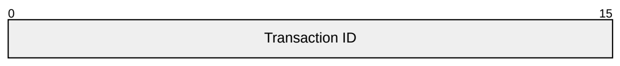


Figure 4.9: Hello Request Message Data

Where:

Transaction ID Is a sequence number uniquely identifying the request.

4.3.4.2. Hello Response Message

The format of the response Management Message Data is:

Where:

Transaction ID Is a sequence number identifying the request for which this is a response.
Status Code Indicates the status of the request.

Code	Meaning
0	Success, the hello was accepted by the peer.

Code	Meaning
1	OTHER. This value is used when there is an error other than listed below or to avoid passing too much information to the peer.
TBD	TBD

Table 4.4: Hello Response Status Codes

If the status code is not success (0) then the Hello Response TLV field must contain the Policy ID and Version Information fields to indicate the policy ID[3] and/or version by which the hello is being rejected.

Hello Response TLVs A series of TLV (type/length/value) structures containing the hello data. The format of a TLV is:

The following TLVs are defined:

Type	Description
0	A NULL TLV. If the TYPE is 0 then the length and value fields are not present (this allows TYPE=0 TLVs to be used as padding).
1	Policy ID Is the ID of the Policy under which the Link or Docking Session was initiated, accepted, or rejected. If a Link or Docking Session was rejected because there was no policy that explicitly accepted it (as opposed to one explicitly rejecting it) then the Policy TLV is not present. **Ed. Note**: The Policy is included in the messages to assist auditing. An implementation SHOULD log that the Link or Docking Session was allowed per the Policy ID in the Hello Response.
2	Version Is the version information of the peer. **The exact content of this field is TBD.** It should indicate A) the implementation version of the peer and B) which ZPR RFC version the peer implements. **Ed. Note**: general field added per request from Mathias.
2	Endpoint Authentication ZPR Address (EAZA). This is the ZPR Address that the Adapter MUST use as the "source address" when communicating with the Auth Service to perform ZPR Authentication.
3	Auth Service ZPR Address (ASZA) The ZPR Address of the Authentication Service. The Adapter MUST send all ZPR Authentication messages to this ZPR Address.

[3]This should not give an attacker any useful information since the policy ID is a simple name conveying no information as to the policy itself.

Type	Description
4	Stream ID Length (SIL) The length of Stream IDs that the sender of the Hello Response wishes the receiver to use when sending ZDP packets with Stream IDs (such as transit packets) to the response sender. Valid Stream ID Lengths are 1, 2, 4, 8, and 16 bytes. All ZDP nodes MUST be able to support all valid lengths. This TLA is optional. The default is 2 bytes for stream IDs going between a Dock and an Adapter and 8 bytes for stream IDs going between a Dock and a Forwarder or between Forwarders.
5 -252	Reserved for future use. These values MUST NOT be used. If a Hello Response contains any of these values it MUST be dropped, the event reported, and the Docking Session or Link terminated.
253	Experimental This is used to experiment with new TLVs. The first byte of the Value contains the Type code point proposed for the TLV. The type is in the range 3-252, inclusive.
254	Vendor Specific. This TLV contains vendor specific (that is, non-standard) TLVs. The first three bytes of the Value contain an IEEE802 OUI identifying the vendor.
255	Locally defined.

Table 4.5: Hello Response TLV types

The Hello Response MUST contain the POLICY ID TLV and VERSION TLV.

If any required TLV is missing then it is considered a handshake failure and the Docking Session or Link is not brought up.

TLVs may be in any order.

A TLV MUST NOT appear more than once in a Hello Response.

4.3.5. Stream ID Request

The Stream ID Request message requests a Stream ID to use in forwarding packets via a specified Visa/Stream be returned.

The ZDP Type field indicates that the message is a Request or Response.

The Request message is a Per-Flow management message. The Stream ID Field of the Request contains an Offered Stream ID. The Offered Stream ID is one that the sender of the Request suggests that the receiver use. The receiver may accept or reject the offered ID. The receiver returns the ID to use, even if it is the offered ID, in the response. See the Stream ID Response in § 4.3.6.

The format of the Stream ID Request Message is:

Where:

Transaction ID Is a sequence number uniquely identifying the request.

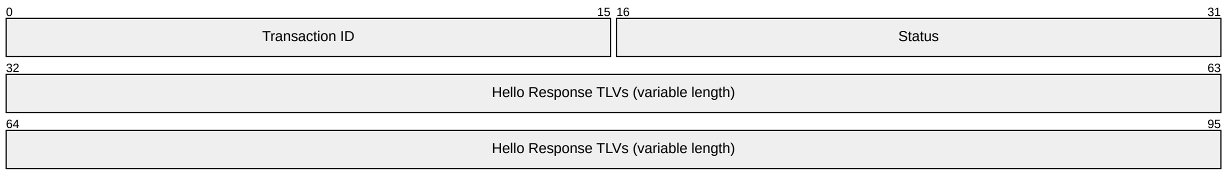

0
15
16
31
Transaction ID
Status
32
63
Hello Response TLVs (variable length)
64
95
Hello Response TLVs (variable length)

Figure 4.10: Hello Response Format

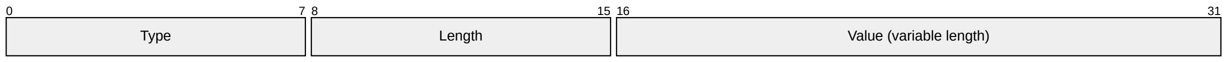


Figure 4.11: TLV Format

First Packet Length If this field is non-0 then the first packet of the stream is being carried in the request. If the first packet of the stream can fit in the request (which is likely if the request a TCP SYN or similar packet) then the upstream MAY include the packet in the request in order to reduce latency. The downstream would then send the packet on in the same manner if the herald operation succeeds. See Figure 12 for the packet structure.

If this field is 0 then the first packet is not present.

Visa&Stream Name Length This is the size of the (following) Visa/Stream Name field.

Visa&Stream Name Is the name of the Visa and Stream for which a Stream ID is being requested.

4.3.6. Stream ID Response

The Stream ID Response is sent in response to receiving a Stream ID Request. It indicates whether the Stream has been provisioned or not. If not, it indicates why. If the Stream is in service, it returns the Stream ID to use by the upstream to send packets under the auspices of the Visa. This ID may be the offered ID or an ID that the sender of the Response selects.

The ZDP Type field indicates that the message is a Stream ID Response.

The Stream ID Response is a Per-Flow message.

The ZDP Stream ID contains the Offered ID from the Stream ID Request. The Management Message Payload contains, among other things the ID the upstream should actually use.

The format of the Stream ID Response is:

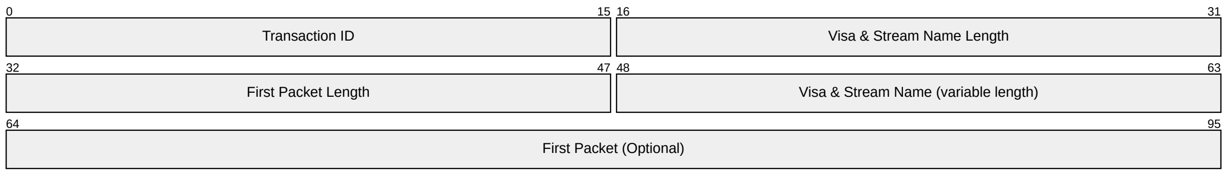


Figure 4.12: Stream ID Request Message Format

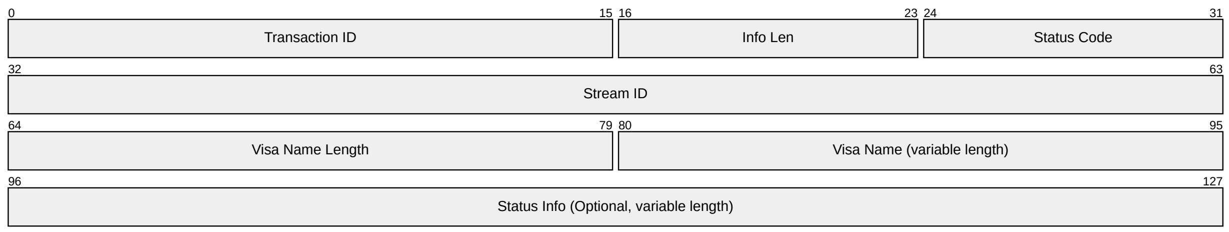


Figure 4.13: Stream ID Response Message Format

Where:

Transaction ID Is the Transaction ID of the Stream ID Request that this message is in response to.

Status Code Indicates the status of installing the Visa:

Code	Meaning
0	Stream provisioned successfully.
1	Other – none of the following or to avoid passing too much information to the peer.
TBD	No Such Name - No Visa/Stream with the given name exists.

Table 4.6: Stream ID Response Status Codes

Info Len Is the length of the Status Information field. It may be 0.

Visa Name Length The length of the Visa Name.

Visa Name The visa's name. This must be the same as in the request.

Status Information Is any additional status information. Typically, this might be a human-readable message designed to assist diagnosis of the issue.

Stream ID If the status is success, then this is the Stream ID that the upstream should use when sending packets under the auspices of the Visa. This is the Final Stream ID. If the status indicates a failure, then this is the Offered Stream ID from the Stream ID Request; the requester uses this to determine which Request has failed and take appropriate action. (Note that the ZDP header always contains the offered ID).

The length of this field is determined by the Hello Response TLV.

If an Upstream Forwarder receives a Stream ID Response from a Downstream Forwarder and the Offered Stream ID in the response contains a Stream ID for a Stream that does not exist (perhaps it was deleted due to some deferred error) then the Upstream MUST silently discard the Response. Further, the Upstream MUST NOT send a ZDP message specifying the non-existent Stream to any other ZPR entity (This avoids packet storms).

As soon as an Upstream Forwarder receives a successful Response it may send ZDP Transit Packets using the Stream ID specified in the response.

4.3.7. Stream ID Withdrawal Request and Response

The Stream ID Withdrawal request message is used when a Node must take a Stream out of service that it had formerly placed in service.

The ZDP Type field indicates that the message is a Stream ID Withdrawal Request.

The Stream ID Withdrawal Request and Response messages are Per-Flow messages.

The ZDP Stream ID contains the Final Stream ID that was used when the Stream was initially heralded.

The format of the Request is:

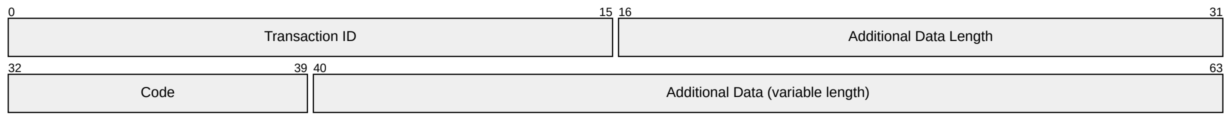


Figure 4.14: Stream ID Withdrawal Request Management Message Data Format

Where:

Transaction ID Is a sequence number uniquely identifying the request.
Code Is a code indicating the reason that the Stream ID is being withdrawn. Typically, this value would be entered in local logs or displayed on a console. Codes include:

Code	Meaning
0	No reason given.
TBD	Egress Dock Session Terminated. Forwarders receiving this code MUST NOT attempt to find alternate routes to the egress.

Table 4.7: Stream ID Withdrawal Reason Codes

Additional Data Length Is the size of the Additional Data field, in bytes. Since Additional data is optional, this field may be 0.
Additional Data Is additional data explaining the reason for withdrawing the Stream ID. Typically, this information would be entered in local logs or displayed on a console. The Node MUST NOT have any expectations as to the content of this field nor may it make any decisions based on the content of the field.

The Stream ID Withdrawal Response is used to indicate that the node received the request and has taken appropriate action.

The ZDP Type field indicates that the message is a Stream ID Withdrawal Response.

The format of the Stream ID Withdrawal Response:

Where:

Transaction ID Is the Transaction ID of the request for which this message is a response.

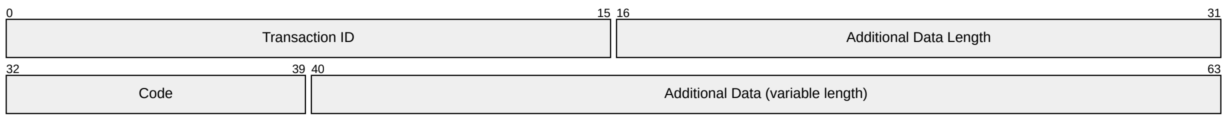


Figure 4.15: Stream ID Withdrawal Response Message Format

Code Indicates the status of the Stream ID Withdrawal:

Code	Meaning
0	Stream ID withdrawn successfully
1	OTHER – error other than those listed below occurred OR to avoid passing too much information to the peer.

Table 4.8: Stream ID Withdrawal Response Codes

Additional Data Length Is the size of the Additional Data field, in bytes. Since Additional data is optional, this field may be 0.

Additional Data Is additional data explaining the reason for retracting the Visa. Typically, this information would be entered in local logs or displayed on a console. The Node MUST NOT have any expectations as to the content of this field nor may it make any decisions based on the content of the field.

If the request specifies a Stream which is no longer in service, a response indicating success will still be issued.

Stream ID Withdrawal Request and Response follow the common Request/Response semantics (§ 4.4).

4.3.8. Register and Deregister Endpoint Addresses Request and Response

The Register Endpoint Addresses Request allows the Adapter to inform the Endpoint of the IP Address(es) of the Endpoint. This is used to inform the ZPR Network's routing so that it can direct Endpoint Packets addressed to the Endpoint to the correct Egress Dock.

Ed Note: Rationale: When an Endpoint is a server (*e.g.*, a DNS server), it normally does not send packets (Endpoint Packets) until it receives a request (the request is also an Endpoint Packet). In order to route these requests to the Endpoint, the ZPR network must know to which Dock the Endpoint is connected (and therefore the Endpoint's IP address). This management message is the mechanism by which the network learns which Dock is serving which Endpoint IP Address.

These messages are not Per-Flow Management messages.

4.3.8.1. Request

The Register and Deregister Requests have the same format.

The formats of the Register and Deregister Request messages is:

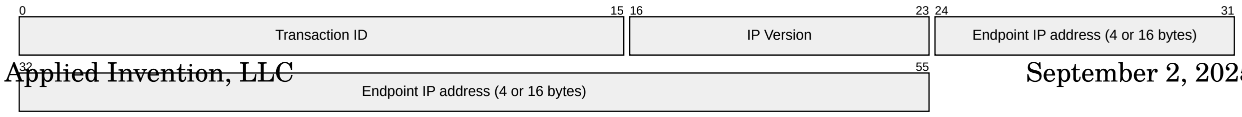

0
15
16
23
24
31
Transaction ID
IP Version
Endpoint IP address (4 or 16 bytes)
32
55
Endpoint IP address (4 or 16 bytes)

Figure 4.16: Register and Deregister Endpoint Address Management Message Data

Where:

Transaction ID Is a sequence number uniquely identifying the request.
IP Version Is the IP version of the Endpoint Addresses. The value of this field MUST be either 4 or 6.
Endpoint IP Address Is IP address of the Endpoint.

4.3.8.2. Response

The response indicates the status of the request.

The Register and Deregister Responses have the same format.

The format of the Register and Deregister Responses is:

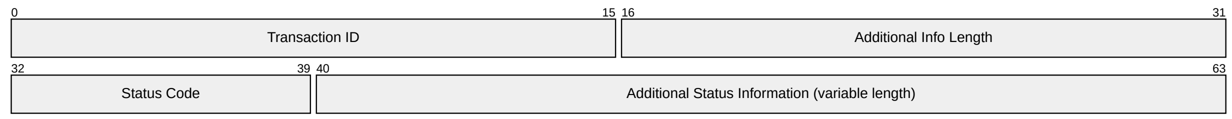


Figure 4.17: ENDPOINT Address Registration and Deregistration Response Packet Format

Where:

Transaction ID Is the Transaction ID of the request for which this message is a response.

Status Code Is a code indicating the status of the Register or Deregister operation. Defined status codes are:

Value	Meaning
0	Success -the registration has been accepted and installed in the network OR the registration is not explicitly allowed by a ZPR policy. In the latter case, the peer MUST A) Report the policy violation and B) install the necessary state to receive packets with the address and silently discard them.
1	OTHER – some error other than those listed below OR to avoid passing too much information to the peer.
	Unsupported address family: Address family is 4 or 6, but that version of IP is not supported
	Length Error (*e.g.*, the IP version is 4 but the two addresses are not a total of 8 bytes long).
	Invalid address family (value is neither 4 nor 6).
	No resources
	Addresses invalid for some reason OTHER THAN policy specifications (*e.g.*, specifying an IP broadcast address)
...	...

Table 4.9: Endpoint Addresses Binding Response Status Codes

Note that there is not a failure code indicating that the registration failed because the network's policies do not allow the address specified in the request. This is because a compromised peer could exhaustively try various addresses to see which are and are not allowed by policy. Such a peer would then gain some knowledge of the policies in the network.

The defined errors all indicate various non-policy error conditions (*e.g.*, lack of network resources or incorrect message formatting) and thus are not vectors for revealing Policies to a potentially compromised Endpoint or Adapter.

Additional Info Len Is the length of the Additional Information field. It may be 0.

Additional Information Is any additional status information. Typically, this might be a human-readable message designed to assist diagnosis of the issue.

4.3.8.3. Processing

The Register and Deregister Endpoint Address Request and Response messages follow the request/response semantics described in § 4.4.

If an Endpoint has multiple addresses it wishes to register, it must do each one separately.

Only Adapters may send a Register Request and only Docks may send a Register Response. Either a Dock or Adapter may send a Deregister Request or Response. These messages may be sent only on a Docking Session. If one of these messages is received outside of these constraints, then that fact MUST be reported. The message MUST be ignored. The receiving node MAY terminate the communications channel with the sender of the message.

Registering the same address multiple times is not considered an error; extra registrations should be responded to indicating success.

Only Adapters may send Register Endpoint Address Request Messages. Only Docks may receive Endpoint Address Request Messages and they must arrive via a Docking Session.

Only Docks may send Register Endpoint Address Response Messages and they MUST be sent via Docking Sessions. Only Docks may receive Endpoint Address Response Messages and they must arrive via a Docking Session.

Unregister Request and Response messages may be sent by either Docks or Adapters.

Either side of a Docking Session may send Unregister Endpoint Address request or response messages. After sending or receiving such a message, the Adapter MUST stop using the specified Endpoint Address to send/receive Endpoint Packets. (A Dock may initiate an Unregister if, for example, policies change and a particular Endpoint Address is no longer considered acceptable to the ZPR Network).

TODO: There are issues surrounding having multiple Endpoints with the same Endpoint Address. The two separate lines of consideration are:

1) If multiple Endpoints get local IP addresses (*e.g.*, 10/8 in IPv4) which is possible now. In theory, the adapter would have to act, a bit, as a NAT device mapping the local address to a global one (global, at least, in the sense of global within a single ZPR Network). This *may* not be an issue of the "Register…". Functionality is limited to servers since servers

tend to have global addresses (needed in legacy IP in order to ensure that the server is reachable from the entire network).

2) As an extension, a service could be distributed over multiple physical machines, each with the same Endpoint Address. In effect this gives anycast. This need not be considered at this point in time.

   Ed. Note: Possible Extension: We might want to include IP Protocol and port information in this message (easily added after the IP address) indicating which protocol/port the server is listening on. Notionally A) the ZPR network could ensure that only Endpoint Packets directed to that address, protocol, port combination are sent to the Endpoint (enhancing security to some degree) and B) providing a way to split services with the same IP address over different physical machines (this 'feels' like a potentially powerful concept/ability to me. We don't have to investigate it right now, but I don't want the thought to get lost either).

4.3.9. Bind and Unbind Endpoint Addresses Request and Response

This section specifies the Bind and Unbind Endpoint Address request and response messages. These messages are used

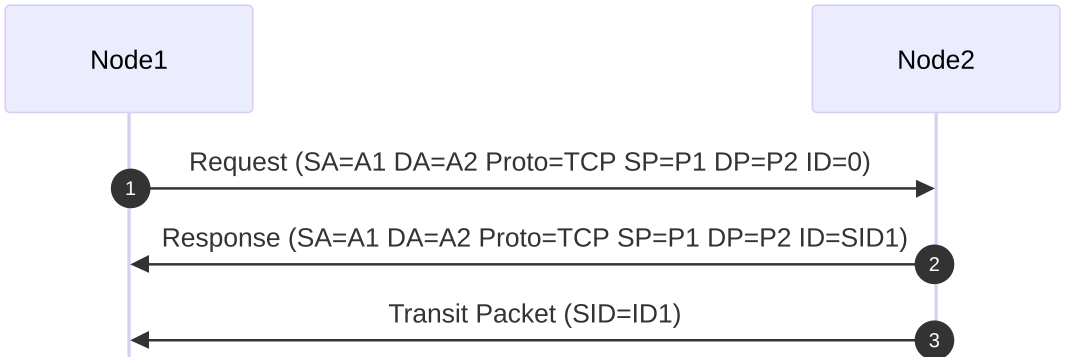
1) By an Ingress Adaptor to request a Stream ID from a Dock to use in transmitting packets of a given Flow into the ZPR Network. If the Dock does not have a Stream ID to use, this will initiate the ZPR Authentication, Visa Granting and Visa Distribution processes.

2) By an Egress Dock to request a Stream ID from the Egress Adapter to use in transmitting packets of a given Flow to the Adapter. In addition, the operation informs the Adapter of the ZPR compression parameters and A2A SA parameters.

THIS NEEDS TO CHANGE, Adapter-to-dock is ok, egress is not, need to explicitly send the sa/da/... from dock to adapter and then get the streamed. Probably make this a new message type!

   Ed Note: Rationale: In order to prevent MTU issues, an Ingress Adapter must remove the IP addresses from packets, leaving room for the ZPR header, and then the Egress Adapter must reinsert the addresses to re-create the original Endpoint Packet. The Bind message is used by the Ingress Adapter to signal the Ingress Dock of the addresses and give a stream ID that the Ingress Dock will use for packets of the communication. These addresses are passed to the Egress Dock as a part of the Visa distribution service. The Egress Dock then performs a similar operation, informing the Egress Adapter of a Stream ID and pair of addresses to use when reconstituting packets tagged with the given Stream ID.

The semantics of the Request are that the requesting node is requesting a Stream ID to use when sending Endpoint Packets with the specified addresses, protocol, and ports to the peer. The response informs the requester of the Stream ID:


Node1
Node2
Request (SA=A1 DA=A2 Proto=TCP SP=P1 DP=P2 ID=0)
1
Response (SA=A1 DA=A2 Proto=TCP SP=P1 DP=P2 ID=SID1)
2
Transit Packet (SID=ID1)
3

Where:

1. Node 1 (*e.g.*, an Adapter) is attempting to send Endpoint Packets from A1:P1 to A2:P2.[4] The Stream id is 0.
2. The other node (*e.g.*, a Dock) evaluates the request and, if acceptable, makes appropriate entries in its forwarding tables. It allocates a Stream ID to use and generates a response. The response has the same addressing, protocol, and port information, but specifies a non-0 stream ID.
3. The requesting node receives the response and updates its tables so that packets being sent to Node 2 with the specified addresses, protocol, and ports are sent with the Stream ID received in the response.

All of these messages are Per-Flow Management messages. The Stream ID in the Bind Request message MUST be 0. For a Bind Response with no error, the ID is the ID the requester must use. If the response is an error, the ID is 0. For the Debind request and response, the Stream ID is the ID that is being removed from service.

The response indicates the status of the request.

The format of the Bind Request message (diagram includes ZDP header and MICV) is:

IS METATADA NEEDED IN THIS REQUEST – TO, FOR EXAMPLE, CARRY A DATA TAG (ALL TRAFFIC IN THIS CONNECTION IS “SECRET”) ... STUFF THAT CAN NOT BE INFERRED FROM THE ADDRESS & PORT INFO. IT CAN ALSO BE USED TO CONTAIN STUFF TO LINK A QUIC UDP HEADER WITH (SAY) THE ORIGINAL HTTP REQUEST WHOSE ALTSVC CAUSED THE QUIC TO COME UP

Where:

Streamid Is set to 0

Packet Count The number of Initial Endpoint Packets in the message. Must be at least 1. Each such packet is, in effect, a separate bind request.

Transaction ID Is a sequence number uniquely identifying the request.

Endpoint Packet Length The length of the Endpoint Packet field.

[4]The Endpoint Packet’s source IP Address is A1 and the source port is P1. The destination IP address is A2 and destination port P2 (we assume TCP or UDP for the sake of discussion).

```
 0                   1                   2                   3
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6 7 8 9 0 1
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+----
|Vers.| ZHSAID  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Common
|     Type      | Excess-Length |         Sequence Number       | ZDP
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ Header
|                            Stream ID                          |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+----
| Packet Count  |
+-+-+-+-+-+-+-+-+
Following repeated Packet Count times:
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|         Transaction ID.       |    Endpoint Packet Length     |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
                  Initial Endpoint Packet
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+---
|                               Pad                             | Common
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+ ZDP
|                               MAC                             | MICV
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+---
```

Figure 4.18: Bind Endpoint Address Request Management Message

Initial Endpoint Packet The Endpoint Packet that started the process. It SHOULD be the entire packet. The sender may send less than the entire packet only if the entire packet will not fit in a single request message; in this case bytes from the end of the Endpoint Packet may be dropped.

The Initial Endpoint Packet does not include any data link headers; it starts with the outer-most IP header of the Endpoint Packet and includes all IP headers, option fields and extension headers, transport headers, and application data.

The Debind request does not have any Management Message Data; the Stream ID in the ZDP header specifies the Stream ID being de-binded.

Multiple responses may be carried in a single packet.

The format of the Bind and Unbind Responses is:

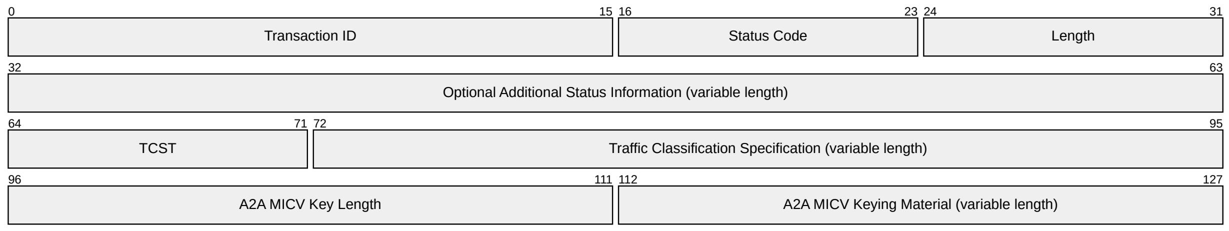

0
15
16
23
24
31
Transaction ID
Status Code
Length
32
63
Optional Additional Status Information (variable length)
64
71
72
95
TCST
Traffic Classification Specification (variable length)
96
111
112
127
A2A MICV Key Length
A2A MICV Keying Material (variable length)

Figure 4.19: Bind and Unbind Endpoint Address Response Management Message Data

Where:

Transaction ID Is the Transaction ID of the request for which this message is a response.

Status Code Is a code indicating the status of the Bind or Unbind operation. Defined status codes are:

Value	Meaning
0	Success - the binding has been accepted and installed in the network OR the binding is not explicitly allowed by a ZPR policy. In the latter case, the peer MUST A) Report the policy violation and B) install the necessary state to receive packets with the Stream ID and silently discard them.
1	OTHER – some error other than those listed below OR to avoid passing too much information to the peer.
2	Not Authenticated The Adapter/Endpoint has not performed ZPR Authentication. Unsupported address family: Address family is 4 or 6, but that version of IP is not supported. Length Error (*e.g.*, the IP version is 4 but the two addresses are not a total of 8 bytes long). Invalid address family (value is neither 4 nor 6). No resources.

Value	Meaning
	Addresses invalid for some reason OTHER THAN policy specifications (*e.g.*, specifying an IP broadcast address).

Table 4.10: Endpoint Addresses Binding Response Status Codes

that there is not a failure code indicating that the registration failed because the network's policies do not allow the binding specified in the request. This is because a compromised peer could exhaustively try various bindings to see which are and are not allowed by policy. Such a peer would then gain some knowledge of the policies in the network.

Info Len Is the length of the Status Information field. It may be 0.

Status Information Is any additional status information. Typically, this might be a human-readable message designed to assist diagnosis of the issue.

TCST Traffic Classifier Specification Type, indicates the format of the classification. Currently, only type 0 ("IP 5-Tuple") is supported.

Traffic Classification Specification The Traffic Classification Specification. The length of this field depends on the type of specification. The format of the classifier is described in fields of the type 0 classification specification are described in § 4.3.9.1.

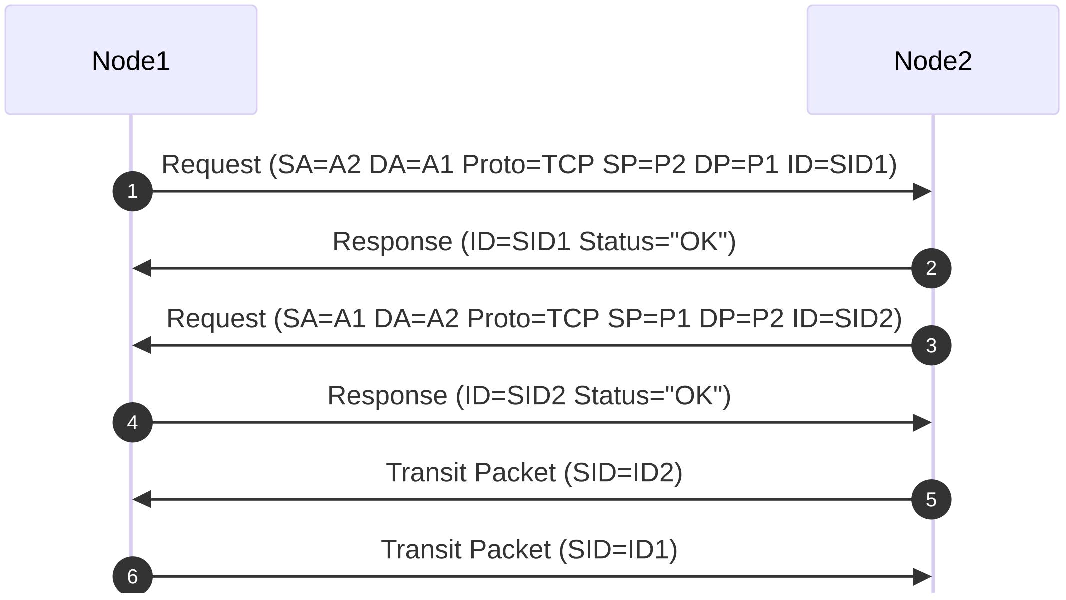
A2A MICV Keying Material Length The length of the keying material, in bytes.

A2A MICV Keying Material This is the keying material for the A2A MICV. The Endpoint uses this to generate the A2A MICV. They keying material is encrypted using a Visa/Endpoint security association. If there are multiple Visa servers, each providing a portion of the keying material, then each portion is encrypted in an SA between the originating Visa Server and the Adapter.

For a successful Bind Response, the ZDP header's Stream ID field specifies the stream ID to use. For a response indicating an error, the Stream ID is 0. For the Debind response, the ZDP header's Stream ID field confirms the Stream ID being removed from service.

The Bind Endpoint Address Request and Response messages follow the request/response semantics described in § 4.4.

The binding operates as follows. Assume a TCP traffic flow between two Endpoints. *Endpoint 1* is on *Node 1*, along with *Adapter 1*. *Endpoint 1* is connected to a Dock on *Node 2*. *Endpoint 1* has Endpoint Address A1 and TCP port P1. Similarly, *Endpoint 2*'s Endpoint Address is A2 and its TCP port is P2. The Bind Endpoint Address requests and responses are:


Node1
Node2
1 Request (SA=A2 DA=A1 Proto=TCP SP=P2 DP=P1 ID=SID1)
2 Response (ID=SID1 Status="OK")
3 Request (SA=A1 DA=A2 Proto=TCP SP=P1 DP=P2 ID=SID2)
4 Response (ID=SID2 Status="OK")
5 Transit Packet (SID=ID2)
6 Transit Packet (SID=ID1)

Where:

1. Adapter 1 (on Node 1) is instructing the Dock (on Node 2) to use Stream ID *SID1* when sending TCP Packets from A2:P2 to A1:P1 to Endpoint 1. These packets are sent as Transit Packets so the actual Endpoint Address and port information has been compressed out. See 5.
2. The Dock confirms the request.
3. The Dock swaps the source and destination Endpoint Addresses, and TCP ports, assigns a Stream ID to use, and instructs the Adapter to use SID2 when sending Endpoint Packets from A1:P1 to A2P2 (again, as Transit Packets so the addresses and ports are compressed out). See 6.
4. The Adapter confirms the request.

Note that the order of operations is notional; steps 2 and 3 could be reversed.

The Dock in Node 2 is informed of the Endpoint Addresses, IP protocol, and port information for Endpoint Packets carried in a particular Transit Packet flow by the Visa setting up the flow.

4.3.9.1. IP 5-Tuple Traffic Classifier

The format of the type-0 Traffic Classifier, the "IP 5-Tuple Traffic Classifier" is:

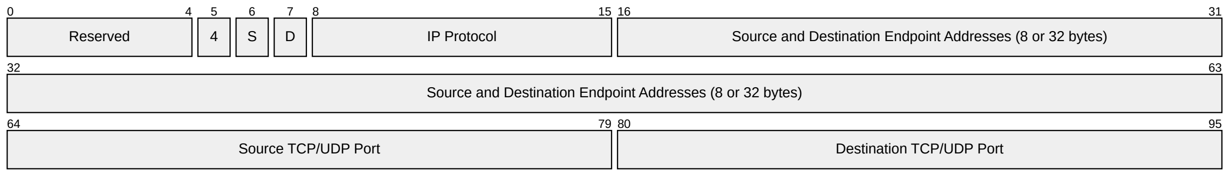

0 4 Reserved
5 4
6 S
7 D
8 15 IP Protocol
16 31 Source and Destination Endpoint Addresses (8 or 32 bytes)
32 63 Source and Destination Endpoint Addresses (8 or 32 bytes)
64 79 Source TCP/UDP Port
80 95 Destination TCP/UDP Port

Figure 4.20: IP 5-Tuple Traffic Classification Spec

Where:

4 "IP V4" if set, the packet must be IPv4 and the Endpoint Addresses 32-bit IP v4 addresses. If clear, Endpoint Packets must be IPv6 and the Endpoint Addresses 128-bit IP v6 addresses.
S If set (1), the source port field is present and packets must match this value to be accepted by the classifier. If not set (0), the field is not present and the source port field of the packets is wildcarded.
D If set (1), the destination port field is present and packets must match this value to be accepted by the classifier. If not set (0), the field is not present and the destination port field of the packets is wildcarded.
IP Protocol Is the IP protocol for packets to be accepted by the classifier.
Source and Destination Endpoint Addresses Are the source and destination Endpoint Addresses of packets to be accepted by the classifier.
Source Port The source TCP or UDP port that packets must match to be accepted by the classifier. This field is present if and only if the "S" bit is present and the IP Protocol is either TCP or UDP.
Destination Port The destination TCP or UDP port that packets must match to be accepted by the classifier. This field is present if and only if the "D" bit is present and the IP Protocol is either TCP or UDP.

4.3.10. Authentication Messages

The Authentication messages carry opaque authentication data between a Dock and Adapter. Data can go in either direction. The Authentication Request transports opaque authentication data from Dock to Adapter, or vice versa. The Authentication Response indicates whether the request message was received and is correct, or not. The response DOES NOT indicate the results of the authentication operation.

The format of the Authentication Request message is:

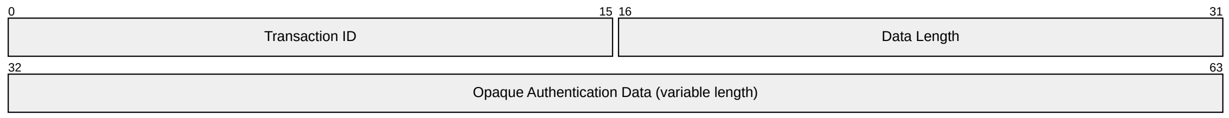


Figure 4.21: Authentication Request Management Message

Where:

Transaction ID Is a sequence number uniquely identifying the request.
Data Length The length of the Opaque Authentication Data, in bytes.
Opaque Authentication Data Is the authentication data. The length of the data is in the length field of the overall packet, see Figure 31.

The format of the Authentication Response message is:

Where:

Transaction ID Is the Transaction ID of the request for which this message is a response.
Status Code Is a code indicating the status of the Request message. Defined status codes are:

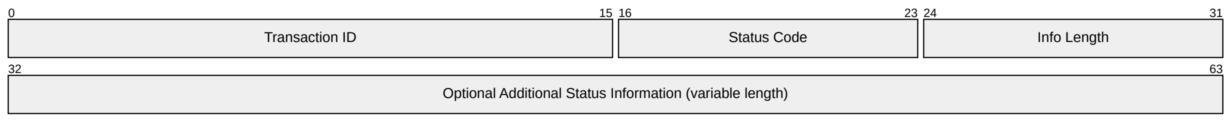


Figure 4.22: Authentication Response Management Message Data

Code	Meaning
0	Success
1	OTHER – some error other than those listed below occurred OR to avoid passing too much information to the peer.
...	Failure

Table 4.11: Authentication Response Status Codes

Info Len Is the length of the Status Information field. It may be 0.
Status Information Is any additional status information. Typically, this might be a human-readable message designed to assist diagnosis of the issue.

The Authentication Request and Response messages use the Request Response semantics described in § 4.4.

4.3.11. Report

The Report packet carries a report message from an Adapter to a Dock to be entered into the ZPR reporting system. The Dock MAY perform filtering, rate limiting, and so on, on Report messages received from Adapters.

This is a Non-Per-Stream message.

The format of the packet is:

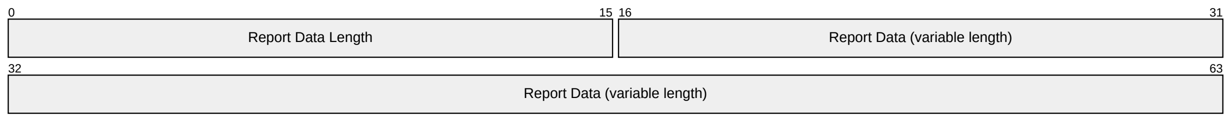


Figure 4.23: ZIP Report Message Format

Where:

Report Data Length Is the length of the Report Data, in bytes.
Report data Is the data to be reported.

4.3.12. Set Path MTU Request and Response

The Set Path MTU request and response messages are bidirectional. They are used

1) When a Dock needs to inform an Adapter of a PMTU to use. The Dock learns of the PMTU from the Visa.

2) When an Adapter needs to inform the Dock of a PMTU on the Adapter-to- Endpoint network. The Dock passes this information upstream, to the ingress Dock, via Path MTU messages.

Set Path MTU is a Per-Flow management operation, the MTU is applicable to the flow identified in the header's Stream ID field.

The format of the Path MTU Request message is:

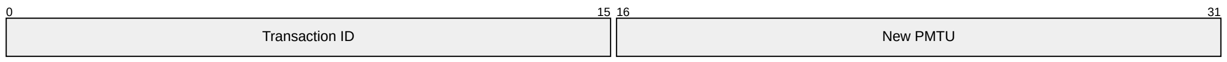


Figure 4.24: Path MTU Request Management Message Data

Where:

Transaction ID Is a sequence number uniquely identifying the request.
New PMTU Is the new Path MTU to use for all packets sent via the Tether.

The message is propagated back to the Ingress Adapter.

The Path MTU Response message is sent in response to a Path MTU Request. It indicates the status of the request. The format of the response is:

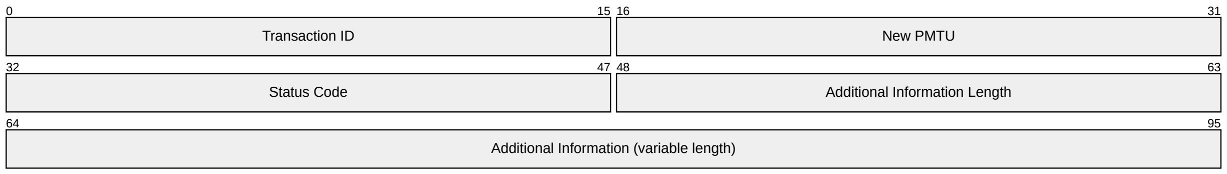

0
15
Transaction ID
16
31
New PMTU
32
47
Status Code
48
63
Additional Information Length
64
95
Additional Information (variable length)

Figure 4.25: Path MTU Response Management Message

Where:

Transaction ID Is the Transaction ID of the request for which this message is a response.
New MTU The MTU that the upstream is using. The PMTU MAY be smaller than the PMTU in the request.
Status Code Indicates the status of the Request. Code values are:

Code	Meaning
0	Success
1	OTHER – some error other than those listed below occurred OR to avoid passing too much information to the peer.
...	Other codes, TBD, they all indicate Failure

Table 4.12: Set Path MTU Request Response Codes

Additional Information Length Is the length of the additional information field. If no additional information is present then this field contains 0.
Additional Information Any additional information the sender may wish to include in the message (for example, it may be a string to include in a logging message).

The Set Path MTU Request and Response messages use the Request Response semantics described in § 4.4.

4.3.13. Destination Unreachable

A Node or Adapter uses the Destination Unreachable message to inform the source Endpoint that a ZDP Transit packet could not be delivered. This message is analogous to ICMP Destination Unreachable messages in IP networks. If the Ingress Adapter receives a Destination Unreachable it MAY fabricate a proper ICMP Destination Unreachable to send to the Source Endpoint.

This is a Per-Flow Message, the Stream ID field of the header identifies the flow from which a packet caused a Substrate Destination Unreachable message. That is, if a ZDP Transit packet is received on Stream *S*, then the Stream Id field in the Destination Unreachable is set to S.

ZDP Destination Unreachables MUST NOT be sent by a Node as a result of receiving an ICMP destination unreachable from a substrate (see § 3.17). If an Adapter receives an ICMP destination unreachable (or similar indication) from the network connecting the Adapter and the Endpoint then the Adapter SHOULD generate an appropriate ZDP Destination Unreachable message to send to the source.

The format of the Management Message Data is:

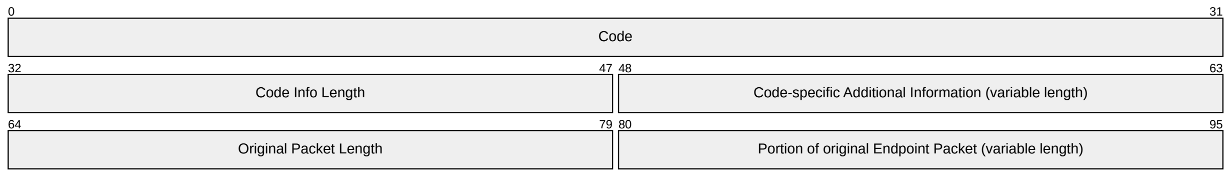

0
31
Code
32
47
48
63
Code Info Length
Code-specific Additional Information (variable length)
64
79
80
95
Original Packet Length
Portion of original Endpoint Packet (variable length)

Figure 4.26: Destination Unreachable Management Message Data Format

Where:

Code Is a code indicating the reason that the packet could not reach its destination:

Value	Description
0	Unspecified. There is no additional Code Specific Information.
TBD	Terminated or Unknown Tether A Dock or Adapter received a packet from the peer with a Tether ID that is unknown or known to be closed. In this case the Code Specific Additional Information field contains the Tether ID.

Value	Description
TBD	Unknown Stream ID. The Code Specific Additional Information Field contains the unknown Stream ID. If the Visa ID is not one that the Node receiving this message has in service for sending Endpoint Packets to the peer, the node MUST silently discard the message. It SHOULD report this to the admin service as a potential attack. If a Forwarder, Dock, or Adapter receives this message from downstream it SHOULD remove any entries in its lookup and forwarding tables that could result in sending messages downstream with the specified ID. **Ed. Note**: I could have had the downstream send a “delete visa” message to the upstream. I decided to send an “unknown id” instead since it separates the detection of the unknown ID (by the downstream) and the decision of what to do about it (by the upstream). For example, in this model the upstream might decide to reherald the visa rather than do the delete.
TBD	Path MTU Exceeded This message is generated by a Forwarder Egress Adapter, or Egress Dock in the event that a ZDP Transit packet is too large for the next hop’s Path MTU[5]. It is sent upstream toward the Ingress Adapter. The Ingress Adapter then performs the correct ICMP operation (*e.g.*, sending a path MTU Exceeded ICMP message to the Endpoint). The Code Specific Additional Information Field contains the suggested Path MTU.
TBD	Destination Endpoint Unreachable The egress adapter can not reach the destination Endpoint for some reason.

Table 4.13: ZIP Destination Unreachable Reason Codes

Code Info Length The size of the “Code Specific Additional Information” field, in bytes.

This field may contain 0, in which case no Code Specific Information has been added to the packet.

Code Specific Additional Information Information that is specific (that is amplifies) the code. If the Code Info Length field’s length is 0 then this field is always 0 bytes long. If the Code Info Length field’s length is non-0 then the maximum length of this field is the size of the data item being conveyed in this field (*e.g.*, if it is a Stream ID then the Configuration-specified Stream ID size).

Original Packet Length The size of the “Original Endpoint Packet” field, in bytes.

Portion of Original Endpoint Packet Is a portion of the packet. This should be as much of

[5]While the Admin Service should know the path MTUs and be able to inform the forwarders, etc., of the current MTU, this might not occur when the substrate over which a Link or Docking Session operates is an IP network.

the original packet as is available to the Node generating the message. Typically, this is what is returned in the ICMP message reporting the condition).

A Node SHOULD implement mechanisms to limit the rate at which these messages are generated in order to mitigate the effect of attacks designed to flood nodes with these messages.

4.3.14. Configuration Request & Response

TBD:

Message sent after the link comes up

Exchanges various bits of config info.

This looks to need to be free form, tlv stuff

Mathias currently has

Raft id for joining state database

Visa service address (zin)

Visa service name (for use with tls)

Visa service visa (visa to use when talking with the visa service)

Priobably will need others.

Grr.

This can be big. . . .

4.3.15. Initiate Authentication Request and Response

The Initiate Authentication message is used by the Dock to direct an Adapter to start the ZPR Authentication process on behalf of the Endpoint.

The format of the request message is:

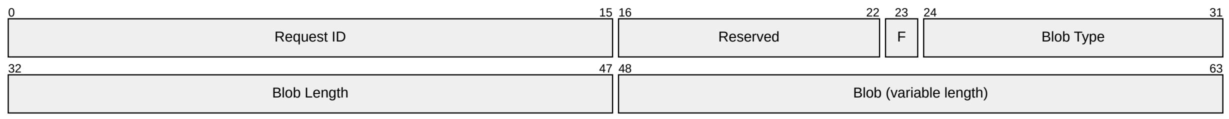


Figure 4.27: Initiate Authentication Request

Where:

Request ID Is a sequence number that uniquely identifies the Request.
0000000 7 Bits of 0.
F Flag Bit, If this bit is 0 then bootstrapping is not enabled and the rest of the payload (type and length) MUST be 0. If this bit is 1 then bootstrapping is enabled and the reset of the payload is valid.
Blob Type Is the authentication payload Blob type.
Blob Length Is the length of the Blob, in bytes.

The following blob types and formats are defined:

Blob Type 0 This is the "null" blob, to be used for testing and debugging. It should never be used in a production environment. This blob is empty; the Blob Length MUST be 0.

Blob Type 1 This is a nonce/timestamp/HMAC blob. Its format is:

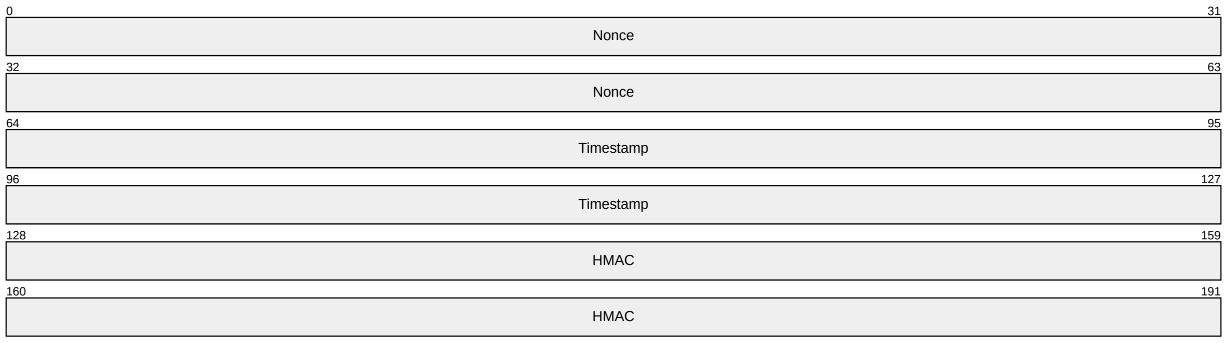


Figure 4.28: Type 1 Blob Format

Where:

Nonce Is a 64-bit nonce. **DETAILS TBD: WHEN UPDATED, HOW, ETC**

Timestamp Is a 64-bit timestamp that is the number of seconds since 00:00:00 01/01/1970, UTC (conveniently, the value returned on unix/linux sytems by the time() system call).

HMAC A 64-byte HMAC calculated over **WHAT?**

The format of the response is:

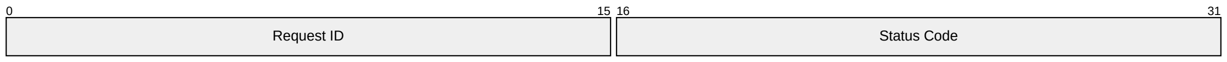


Figure 4.29: Initiate Authentication Response

Where:

Request ID Is the Request ID of the Request for which this is a response.

Status Code Indicates the status of the request:

Code	Meaning
0	No Error: The Initiate Authentication Request has been received, and the authentication procedure will begin. **NOTE WELL**: this does not mean that authentication has completed successfully.
TBD	Errors … **TBD**

Table 4.14: Init Authentication Response Codes

4.3.16. Acquire ZPR Address Request and Response

The Acquire ZPR Address Request is used by the Adapter to request a ZPR Address for the Endpoint. The response is used to acknowledge receipt of the request or indicate that the request failed for some reason.

The format of the Request is

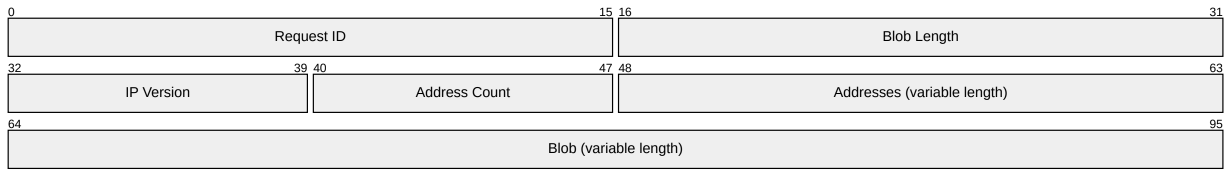


Figure 4.30: Acquire ZPR Address Request Management Message Format

Where:

Request ID Is a sequence number that uniquely identifies the Request.
Blob Length Is the length of the Blob.
IP Version Is either 4 or 6.
Address Count Number of preassigned IP addresses included in the request. If no addresses have been assigned, then this field will be 0.
Addresses IP Addresses that have been preassigned to the Adapter. The Dock MAY wish to use these addresses (if so, it will return them in the Grant message). If addresses have not been preassigned then this field is empty (and AddressCount will be 0).
Blob The Blob received when authentication finishes (step 15 in § 3.26).

ED. NOTE: Should this be something like a SHA of the blob, plus some secret, proving that the requester has the blob, without actually revealing the blob? This can help protect against compromised nodes – I think.

The format of the response is:

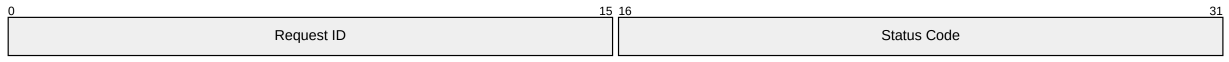


Figure 4.31: Acquire ZPR Address Response

Where:

Request ID Is the Request ID of the Request for which this is a response.
Status Code Indicates the status of the request:

Code	Meaning
0	No Error: The Request has been properly received and will be acted upon in due time. **NOTE WELL**: this message simply acks (or rejects) the request; it does not contain the granted address.

Code	Meaning
TBD	Errors . . . **TBD**

Table 4.15: Acquire ZPR Address Response Codes

4.3.17. Grant ZPR Address Request and Response

The Grant ZPR Address Request and Response messages are used by the Dock to give ZPR Addresses to the Adapter/Endpoint. The Dock sends the address to the Adapter via the Grant ZPR Address Request. The Adapter uses the Grant ZPR Address Response acknowledge receipt of the request or indicate that the request failed for some reason.

The format of the Request is

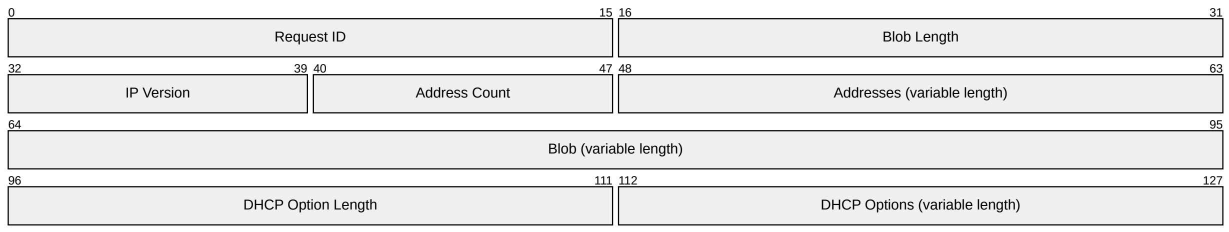


Figure 4.32: Grant ZPR Request Management Message Format

Where:

Request ID Is a sequence number that uniquely identifies the Request.
Blob Length Is the length of the Blob.
IP Version Is either 4 or 6.
Address Count Number of preassigned IP addresses included in the request. If no addresses have been assigned, then this field will be 0.
Addresses IP Addresses that are assigned to the Adapter.
Blob The Blob received when authentication finishes (step 15 in § 3.26).

> **ED. NOTE:** Should this be something like a SHA of the blob, plus some secret, proving that the requester has the blob, without actually revealing the blob? This can help protect against compromised nodes – I think.

DHCP Option Length The length of the DHCP Option field (below). (**Ed. Note:** This is a place holder as we believe that the parameters assigned via DHCP will also want to be assigned via ZPR; this is the mechanism we think will do it).
DHCP Options DHCP Options[6] to be configured in the Endpoint. The details for this are TBD. (**Ed. Note:** This is a place holder as we believe that the parameters assigned via DHCP will also want to be assigned via ZPR; this is the mechanism we think will do it).

The format of the response is:

Where:

[6]Droms (1997)

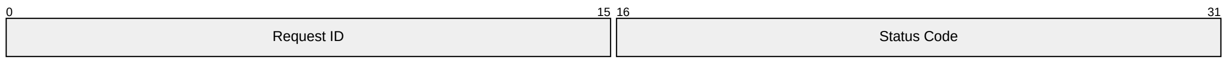


Figure 4.33: Grant ZPR Address Response

Request ID Is the Request ID of the Request for which this is a response.
Status Code Indicates the status of the request:

Code	Meaning
0	No Error: The Grant Request has been properly received and accepted.
TBD	Errors … **TBD**

Table 4.16: Grant ZPR Address Response Codes

4.3.18. Withdraw ZPR Addresses

TODO: NEED TO ADD PROTOCOL MECHANISM FOR THE DOCK TO REMOVE AN ADDRESS. PROBABLY THE SAME PACKETS AS THE GRANT, BUT THE SEMANTIC IS "REMOVE THESE ADDRESSES"

4.3.19. Experimental and Private Use

These messages are for experimental and private use. If an implementation receives such a message that it does not understand it MUST silently discard the message. It MAY log the reception of the message.

4.4. Request/Response Semantics

Many message exchanges require request/response semantics. One peer sends a request of some form and then waits for a response. If a response is not forthcoming, it may either retransmit the request or take some other reporting or error recovery action. Since these semantics are common for all protocols, they are documented once.

The two parties to the Request/Response are termed the requester and responder. The requester initiates some action; the responder generates the response.

Transaction ID numbers are used to match responses to requests, verifying that a request was received and associating any information in the response with the request (*e.g.* a "status code" in a response to a requested operation).

The Transaction ID number space is finely divided. The numbers are per-link/docking session, per message type, and per stream ID (for per-stream messages).

Transaction IDs may wrap.

Transaction ID numbers must be used in sequence, with no gaps.

Sequence numbers start at 0.

The request/response semantics have a window of 1; that is, a request with Transaction ID number *n* MUST receive response from the peer, acting as a positive acknowledgement before a new request with sequence number *n+1* is sent.

Requests and responses that are out of the window are counted and MUST be silently discarded.

The requester handles the following events:

- Request Transmission:
    1. The requester generates the request message.
    2. The message is assigned the next Transaction ID number to send for the message-type, Stream ID, and Link or Docking Session.
    3. Enough information MUST be retained so that the original request, including the Transaction ID number assigned to the request, can be re-transmitted, should the need arise.
    4. The `next_Transaction_ID` number is incremented.
    5. The request is transmitted.
    6. A Response Timer is set to 1 second (this is a default value, it may be overridden by Policy or by circumstances) and started.
    7. A retransmit count is set to 3 (this is a default value, it may be overridden by policy or circumstances).
- Response Timer Expires
  If the response timer expires it indicates that the response has not been received. Either the response is retransmitted, or the request is declared to have failed:
    - If the retransmit count is 0 then the request is deemed to have failed and appropriate action taken. At the very least
        1. The failure is counted
        2. The failure is reported
        3. Further action is taken as described for the specific request.
    - If the retransmission counter is not 0
        1. The Request message is retransmitted. The message contains the same Transaction ID number as the original request.
        2. The response timer is set to 1 second and restarted.
        3. The retransmission counter is decremented.
- Receive Response
    1. Responses contain the Transaction ID number of the request to which they are responding. If a response cannot be matched up with the Transaction ID number of the request that is awaiting a response, then an error has occurred and
        a. The response is counted,

   b. A report detailing the response is generated,
   c. Action as specified for the specific request/response pair is taken.
2. The response's request Transaction ID number matches the sequence number of an outstanding request. The response timer is cancelled.
3. The response is acted upon as described for the specific request/response.

4.4.1. Responder

The responder acts only in response to receiving a request. The responder

1) Processes the request, including checking to make sure that the request's Transaction ID number is the expected Transaction ID number, If so, the expected Transaction ID number is incremented.
2) Generates a response,
3) Puts the request's Transaction ID number in the response,
4) Sends the response.

The request/response semantics implicitly guarantee ordering because only a single request can be outstanding at a time.

When a response to a request is received, the software initiating the request SHOULD be informed of the number of re-transmissions required and the latency between the request and response.

4.4.2. Performance Considerations

ARQ is a simple algorithm and can be performance limited. We chose it because A) the operations which use it are low-frequency operations which are not very performance critical and B) its simplicity minimizes the possibility of implementation errors, pathological anomalies, or security vulnerabilities cropping up. If the performance becomes an issue, then this algorithm could be changed to something else, such as TCP's sliding window.

5. Discussion of Options

5.1. Use of IPv6 Header Extensions

The ZPR protocols could have been implemented as IPv6 header extensions (similar to, for example, IPsec's ESP). This option is attractive in that it would fit ZPR into the IP architecture using mechanisms that were explicitly designed into IPv6 for just this purpose. This could mean, for example, that implementing ZPR's data transport mechanisms would be easier and build on an existing, debugged, and deployed foundation.

The primary counter argument is that a goal of ZPR is to run over "bare datalinks" such as point-to-point fiber optic links and Ethernets. In this case, there is no IP so using the IPv6 header extension mechanism would not be possible. We could deploy IP on these links, but that conflicts with the basic goal (especially since once IP is deployed, other uses would start to crop up).

A second, minor, argument against using the IPv6 header extension mechanism is that it requires allocation of an IP type code point. The IP type is an 8-bit value and well over ½ of the values have been allocated. We could expect pushback from IANA since we could also encapsulate ZDP packets in IP/UDP packets (using a UDP port number, which is not as limited a resource, especially if the port number to use on any given Substrate network is declared in the configuration).

5.2. IP Fragmentation and Path MTU Considerations

Nominally, ZPR does not do fragmentation. In theory, the Admin Service knows the MTUs of all the Links & Docking Sessions and can set the Path MTU accordingly (via the Visas). However, there are three issues that complicate matters:

1) How does the PMTU information get to the Endpoint?

2) What do we do if an Endpoint sends packets that exceed the MTU?

3) What happens when a Substrate Network is IP and the MTU of the substrate changes?

5.2.1. Informing the Endpoint

The PMTU information must get to the Endpoint via an appropriate ICMP message. This is dictated by the fact that the Endpoint/Adapter interface usually is an IP network. The ICMP message would be created by the Endpoint's Adapter. Alternatively, if the Endpoint and Adapter are resident on the same machine, the Adapter could reach in and update the machine's IP tables (probably the routing table, but this is implementation specific) so that the Endpoint will send out packets of an acceptable size (this approach is implementation-specific and will be considered no further).

This begs the question of how the Adapter gets the information needed to create the ICMP message. Policies do not get distributed to Adapters, so the "Admin Service does it via Policy" solution is not available. The approach that seems to work is:

1) Have the Admin Service inform the Dock of a path's PMTU. This would be done when the Dock obtains a Visa for a flow.
2) The Dock, via the Set Path MTU message (§ 4.3.12), informs the Adapter of the PMTU. The Adapter then can evaluate packets sent by the Endpoint and generate ICMP messages back to the Endpoint. There is then the question of to what the PMTU applies:
   a) Everything sent by the Endpoint,
   b) Everything sent by the Endpoint to a specific destination Endpoint, or
   c) Everything sent by the Endpoint that the Adapter will map to a specified Tether.

   The right answer seems to be C since, theoretically, packets sent via different Tethers to the same Destination Endpoint could follow different paths through the network. This in turn implies that the right place to set the MTU is not via policy but rather when the Visa is sent to the Dock. At that point, the Dock can send a message to the Adapter, who can then inform the Endpoint as needed.

5.2.2. Endpoint Sends Packets That Are Too Big

If the Endpoint sends IPv6 packets or IPv4 packets with the DF (Don't Fragment) bit set then the Endpoint's Adapter will intercept the packets, evaluate them against the PMTU for the Tether that the packets would traverse, and, if the packets are too big, drop the packets and send an appropriate ICMP message back the Endpoint. This is the normal and expected IP behavior.

This does not cover the case of the Endpoint sending IPv4 packets with the DF bit clear. In this case, the Endpoint would expect that the packets will be fragmented if needed. Sending an ICMP message back to the Endpoint might work, but then again, it might not.

Options for addressing this are

1) Ignore it
   If a ZDP packet reaches some point where it exceeds the MTU, it is silently dropped. This might even be done at the Ingress Adapter. As an implementation strategy, we might adopt this since fragmentable IPv4 packets are probably fairly rare these days. However, as a protocol architecture/design matter, we should have a better solution. This option is rejected.
2) Have the Ingress Adapter (or any other point in the network) fragment Endpoint IPv4 packets with DF=0 if needed. In this option, whenever a ZDP packet is too big to traverse the next hop Link or Docking Session, the IPv4 packet carried in the ZDP Transit Packet is extracted from the ZDP packet, fragmented per the normal IPv4 fragmentation rules, and then sent to the next hop in two or more ZDP packets:

   gmentation of IPv4 Endpoint Packet (EP)} \end{figure}

   Since the inner Endpoint IP packet is fragmented, it is reassembled by the destination Endpoint.

   This scheme has the large advantage that it most closely mimics IPv4's behavior. The big downside is that an Endpoint, seeing that it is connected to a network with a particular MTU (say 1500 bytes on an Ethernet) is likely to use that MTU. Every one of these packets

```
+--------------+                      +--------------+
| ZDP Hdr      |                      | ZDP Hdr      |
+--------------+                      +--------------+
| EP  IPv4 Hdr |                      | EP IPv4 Hdr  |
+--------------+                      +--------------+
| Endpoint     |- - - - - - - - - - ->| Endpoint     |
|  Packet      |- - - - - - - - - - ->|  Packet      |
|   IPv4 Data  |- - - - - - - - - - ->|   IPv4 Data  |
|              |- - - - - - - - - - ->|    (some)    |
|              |- - - - - -           +--------------+
|              |- - - - -   \
|              |- - - -  \   \        +--------------+
|              |- - -  \  \   \       | ZDP Hdr      |
+--------------+     \  \  \   \      +--------------+
                      \  \  \   \     | EP IPv4 Hdr  |
                       \  \  \   \    +--------------+
                        \  \  \    - >| Endpoint     |
                         \  \  - - - >|  Packet      |
                          \   - - - ->|   IPv4 Data  |
                            - - - - ->|    (rest)    |
                                      +--------------+
```

Figure 5.1: Fragmentation of IPv4 Endpoint Packet (EP)

would end up fragmented by the Adapter since 1500 is also the likely broader network physical MTU, so every Endpoint packet would need to be fragmented (and reassembled). This

a) Could take significant resources at the adaptors,

b) Reduce network reliability (and, as a result, throughput), and

c) Increase the number of small packets on the network (which disproportionally consume resources).

Alternatives are equally unattractive (for example, developing some form of reliable fragmentation protocol would still place large burdens on network components, would not decrease the number of small packets on the network, and would likely increase latency through the network, with concomitant goodput changes).

5.2.3. IP Substrate PMTU Change

Changes to the PMTU in non-IP substrates all can be handled by the Admin Service using existing ZPR mechanisms.

However, wherever there is an IP substrate, it is possible that the IP network's PMTU could change without the Admin Service being aware of it. This could occur in

- A Link's Substrate, between two Forwarders,
- A Docking Session's Substrate, between a Dock and an Adapter,

- The legacy IP network between an Adapter and Endpoint.

Changes could occur for a number of reasons such as an administrative change to a router's configuration or routes changing so that the path between two ZPR Nodes now has a smaller MTU.

If an IP Substrate PMTU changes, it will first become apparent at the ZPR entity adjacent to the Dock Session or Link where the PMTU has reduced (that is, the transmitter of the packet that was too big). Some IP router in the substrate will send an appropriate ICMP "Packet Too Big" message back to the Forwarder, Dock, or Adapter that sourced the packet (the source of that packet). NOTE WELL: From the substrate's perspective, the source of the packet is the Dock, Adapter, or Forwarder that transmitted the packet over the substrate with the reduced PMTU.

The question is then what happens next. There are at least two possibilities:

1) The Forwarder or Dock informs the Admin Service of the change. The Admin Service would then have to send out revisions to existing Visas that the change effects.
2) The Forwarder or Dock could inform the Ingress Adapter of the new MTU. The Adapter would then handle it as described in the previous two sections. The Dock would then have two different PMTUs for the visa - the PMTU received with the Visa and the one received from some downstream IP Substrate. The Dock would instruct the Adapter to use the smaller of the two.

The first option is necessary since any new Visa whose traffic will use the path with the changed PMTU must get the new PMTU information. However, since the Admin Service would be a choke point in the flow of this information, it would be advantageous for the direct notification of option 2 to be used to alter the PMTU for existing Visas.

NOTE WELL: If the Substrate is IPv4 then the Substrate will fragment packets, if needed, to get past a reduced MTU. This fragmentation and reassembly would be transparent to the ZPR Nodes/etc. using the Substrate. ZPR Nodes will ALWAYS send packets over an IPv4 substrate with the DF bit set (preventing fragmentation/etc.).

5.2.4. SUMMARY

- Admin Service is presumed to know the PMTUs. It includes this with the Visa when the Visa is sent to the Dock.
- The Dock informs the Adapter of the PMTU via a Set Path MTU message. This message is sent when the Dock receives the PMTU with the Visa OR when the Dock receives an updated PMTU via a Set Path MTU (§ 4.3.12) message from a Forwarder or Egress Dock.
- If a Forwarder or Egress Dock receives an IP ICMP Too Big message from an IP substrate, it sends a Set Path MTU message upstream, towards the Ingress Dock.
- Too Big IPv6 packets and IPv4 packets with the DF bit set are dropped wherever it is detected that they exceed the PMTU. If dropped at a Forwarder, the Forwarder sends a Set Path MTU message upstream. If dropped at a Dock, the Dock sends an appropriate Set Path MTU message to the Adapter. If dropped at the Adapter, the Adapter sends the appropriate ICMP to the Endpoint.

- If an Endpoint IPv4 packet with DF=0 would exceed a Link or Docking Session's MTU, the Endpoint packet is removed from the ZDP packet, fragmented per the normal IPv4 rules, and then sent via multiple ZDP packets. They are reassembled at the destination Endpoint.

5.3. ZARP

ZDP uses its own protocol to resolve ZPR addresses, ZARP, to find Ethernet addresses instead of using ARP[1] or IPv6 Neighbor Discovery (IPv6 ND)[2]. There are several reasons for this:

1) IPv6 ND requires the use of IP. In theory ZPR does not require that IP be present. Further, IPv6 ND by definition resolves IPv6 addresses. There is no technical requirement that ZPR addresses be IPv6. They could be IPv4 addresses, strings, and so on.
2) As a practical matter, ARP is an IPv4 protocol. If IPv4 is not present in the implementation, then ARP would not be present. ZPR does not require that IPv4 be present, so it cannot rely on ARP.
3) If ARP was extended with a new address family for ZPR addresses, we would need to modify existing ARP implementations to check for the new ZPR address type and pass it to any ZPR code. In most operating systems, ARP is a kernel function, so this would need to be a kernel modification (which would not be possible in proprietary operating systems).

Thus, it seems that purely as an implementation strategy, it would be more efficient to implement a separate ZARP protocol.

The packet headers, etc., of ARP or ND could have been used for ZARP, but ZARP does not need the full generality of those protocols, so it seemed better to simplify the protocol to be specific to ZPR.

5.4. Errors and The Robustness Principle

The Robustness Principle implies that "harmless" protocol errors NOT have dire effects (the "be[ing] liberal in what you accept" part of the principle). An alternate philosophy is that such events represent one of 1) a misconfiguration, 2) a bug in either the code or the specifications, or 3) an attack. In such circumstances it is thought better to make the problem unignorable – *e.g.*, bringing down a Link or Docking Session.

As a general rule we adhere to the later philosophy in this specification – when operational errors are detected, something unrecoverable should happen.

5.5. Exposure of Information

Many of the management messages send information as to the state of the ZPR network to a peer (*e.g.*, status responses to requesters). If the peer has been compromised, this represents a leak of information to an attacker. Since a fundamental goal of ZPR is to provide a secure network, this leak is problematic.

[1] Plummer (1982)
[2] Simpson et al. (2007)

Initially, we thought about simply removing all management messages that conveyed such information. Instead, we would

1) Replace them with a generic "acknowledgement" indicating that a request was received (thereby preventing retransmission, etc.),
2) Have the node that would have issued the problematic message to the peer send the information to the admin service, and
3) Have the node that would have issued the problematic message act in a manner such that the requestor could not determine the state of the network.

This seemed to be too broad a brush for several reasons:

1) There may be instances where sending such information does not reveal sensitive network state to an attacker (knowing that state cannot alter an attacker's behavior),
2) Some protocol developers or network designers may prefer to have more information, sacrificing some security for otherwise-better operational behavior, or
3) Passing around this information can help in diagnosing development or operational issues.

As a result, we have 1) Opted to have the sizes of the 'status' fields be controlled by the ZPR Configuration. Setting the size of a status field to 0 would remove the problematic information. For maximum flexibility, we allow the config to independently set the size of each status field of each message and 2) in certain cases, specified that when internal policy might be revealed to a peer, we instead issue a "success" code and then "go silent" with respect to that peer – black-holing all Transit Packets and faking success on all management operations.

There is a third, middle ground, option that can be considered in the future. Status fields have a non-0 size, but they indicate only success or failure.

5.6. The End to End Message Integrity Check Value

There is some discussion over whether the "end to end" MICV should be from Ingress Dock to Egress Dock (D2D) or Ingress Adapter to Egress Adapter (A2A). This MICV is to protect the compressed Endpoint Packet, ensuring that the packet is not misdirected, tampered with in a way that can compromise the security of the ZPR network.

This note places the generation and checking of the end-to-end MICV in the adapters (that is, we use the A2A MICV). This section explains the rationale for this.

The purpose of the E2E MICV is to prevent a compromised forwarder or an intruder on a link from inserting bad traffic by taking a good Endpoint ZDP Packet, removing its payload (the actual Endpoint-to-Endpoint IP packet) and inserting an attack packet of some form. It does this by calculating a keyed message integrity check over the original Endpoint Packet. The keys are known only by the Ingress and Egress. Any change made to the Endpoint Packet after the integrity check is generated will be detected and appropriate action taken.

The two advantages of doing an A2A MICV vs a D2D are:

1) Docks cannot inject bad Endpoint Packets or alter them. With a D2D MICV they could. Similarly, the data is protected crossing the Docking Session; formerly it was not.

2) The computational load of generating and checking the MICV is moved to the Adapter, which has better scaling properties.

Security is not worsened by moving the MICV to the Adapter. If the Adapter is compromised, it can generate bad traffic, manipulate Endpoint Packets, and so on regardless of where the MICV is calculated or checked. The compromised adapter can generate a good A2A MICV for bad data OR send the bad data to the Dock, which will generate a good D2D MICV. Similarly, if the egress adapter is compromised then it can send any data it wants to the Endpoint.

6. Scaling and Performance Considerations

In this chapter we discuss the design choices made with respect to scaling and performance of ZPR networks.

Specifically, for scaling we are primarily concerned with the number of Visas that are in effect in the network at any one time as that directly effects the amount of storage required in Nodes, the size and complexity of lookup tables and algorithms, and in the end, the design of fields in the ZPR data packets that are used for packet forwarding. In addition, we are concerned with the number of Tethers and Adaptors that are served by a given Dock.

For performance, we have broader concerns. First, there is the complexity and performance of the lookup(s) done in relation to forwarding packets. This is related to the scaling concerns of the previous paragraph. In addition, we are concerned with things like

1) The size of ZPR headers and associated reduction in payload space in individual packets,
2) The latency in starting a traffic flow introduced by installing Visas in the network,
3) The processing overhead of ZPR data packets other than the actual lookup.

6.1. Performance

6.1.1. Goals

The general goals for performance are

1) Forwarders should be able to forward packets at "full speed" – that is, be able to forward an infinite stream of back-to-back minimum sized packets without dropping any due to lookup or output congestion.
   a. The packets are all nominal packets (that is, require no exceptional processing),
   b. The output Link is presumed to be uncongested (that is, the output queue for the Link's interface does not get full), and
   c. There is no particular network service for the traffic that would cause some packets to be dropped (*e.g.* a discipline such as RED (Floyd and Jacobson 1993)).

   This is essentially the same as common performance test for IP routers and switches.
2) The packet overhead of the ZDP packet headers, when running on a datalink substrate (such as raw Ethernet) should be 0 bytes for IPv6 Endpoint Packets and no. more than 24 bytes for IPv4 Endpoint Packets. When running over an IP/UDP substrate, the size of the IP and UDP headers (28 and 48 bytes for IPv4 and IPv6, respectively), can be added to this.
3) Overall, there should be no more than a 5% reduction in per-packet payload.

Ed. note: These goals were laid out in one of the early architecture/requirements documents. Can of course be debated/changed.

6.1.2. Forwarding Considerations

1) The main forwarding decisions taken by a Forwarder are based on looking up the Stream ID field in the ZDP Transit Packet header (§ 4.2.5). This field is local to each Link, is unidirectional (that is, it applies only to traffic from Forwarder *A* to Forwarder *B* and not traffic from *B* to *A*) and is assigned by the Forwarder that will *receive* the traffic. This allows the receiving forwarder to assign IDs in a manner that is optimal for that forwarder's lookup implementation. For example, the ID could be the actual address in memory of the data structure describing how to handle the packet or it could be an index into an array of similar information.

   We note that the Stream ID is the same size as an IPv4 address and that there are many algorithms and data structures that support high-speed lookups of IPv4 addresses. Thus, existing IPv4 lookup techniques should also be adequate for looking up Visa IDs.

2) IP packet lookups often entail many different fields. For example, core routers will often lookup the source address of a packet in order to do reverse path filtering. Others will do a lookup on additional fields (such as the IP protocol number and TCP or UDP port numbers) to do more sophisticated packet filtering. ZPR eliminates this by doing each lookup solely on the Visa ID.

3) A ZPR Dock will verify that Endpoint packets received from Adapters have the correct Endpoint Addresses and (possibly) IP protocol and TCP/UDP port numbers. This is a relatively straight-forward compare operation and should require significant resources.

6.1.3. Packet Processing Considerations

ZDP Transit Packets require more processing that IP packets since they are cryptographically protected on each Link. Thus, they require decryption and re-encryption (and verification & recalculation of the MICV) as they are processed by a Forwarder.

These cryptographic operations represent a significant processing cost. This is necessary, however, to provide the security properties of ZPR.

In addition, we note that modern encryption algorithms, such as AES and SHA, are designed for performance, on both software and hardware. Some modern outboard security processors report 100Gbs or more performance, so we are optimistic that the security protocol performance cost will not be excessive.

Since only the ZDP header is subject to per-hop encryption/decryption, etc., the overall cost is further contained.

6.1.4. Packet Size Considerations

The size of the ZDP header is controlled by the ZPR Configuration. The Configuration specifies the sizes of many fields in the ZDP header.

A goal is that, for ZDP running over a raw substrate (that is, not an IP/UDP substrate), and for IPv6 Endpoint Packets, the resulting packet header should not reduce goodput at all, nor should the addition of the ZDP header enlarge the packet so as to cause a potential MTU issue. This is accomplished by removing the IPv6 Addresses from the Endpoint Packet at the Ingress Adapter and reclaiming that space for the ZDP header. The Egress Adapter reconstitutes the original IPv6 header (reinserting the addresses). Thus the ZDP header cannot be more than 32 bytes long.

This goal does not apply to IPv4 for three reasons: 1) IPv4 is a legacy protocol and presumably its use is declining, so tuning ZDP to IPv4 does not make a lot of sense, 2) IPv4 supports fragmentation and can more easily handle potential MTU issues, and 3) there are not enough bytes in the IPv4 header that can be reclaimed for use (only 8 address bytes, with perhaps another 4 that could be used).

The ZDP header fields, and their lengths, are:

Field	Size, in bytes
ZPI	1
Type	1
Excess Length	1
Sequence Number	2
Stream ID	4
Padding	0 to 8
ZDP MAC	4
A2A SAID	1
A2A MAC	4
Total	18 to 26

The padding is given as either 0 or 8 bytes (with a total overhead of either 19 or 27 bytes) on the assumption that the encryption block size is either 8 or 16 bytes.

This meets the target for packet size for IPv6. In fact, the packet will shrink by 5 or 13 bytes.

For IPv4, there are 11 or 19 bytes in excess of the reclaimable space. Thus, the MTU for IPv4 packets would need to be reduced by that amount. Assuming a 1500 byte MTU (for Ethernets), the additional bytes of the ZDP header represent a 0.7% to 1.2% reduction.

6.1.4.1. IP/UDP Substrate

If ZDP is running over an IP/UDP substrate then, in addition to the ZDP header, an additional IP/UDP header must be prepended to the packet. This requires either 28 bytes (for an IPv4 substrate) or 48 bytes (for IPv6). This can only be accommodated by an appropriate reduction in the MTU – It cannot be compensated for by compressing the Endpoint Packet's headers.

For IPv4, the total overhead is 39 bytes (28+11) or 47 bytes (28+19) – 2.6% or 3.1% of a 1500 byte MTU, respectively.

For IPv6, the total overhead is 43 bytes (48-5) or 35 bytes (48-13), 2.9% or 2.3% of a 1500 byte MTU, respectively.

An IPv4/TCP Endpoint packet on a 1500 byte Ethernet has:

- 14 Byte Ethernet Header
- 20 Byte IPv4 Header
- 20 Byte TCP header

Leaving 1446 bytes of payload.

When adding ZDP over IP Substrate headers, the available payload is:

ZDP Encapsulation	Overhead	Payload	Percent of NonZPR
IPv4 0 Pad	39	1407	97.3%
IPv4 8 Pad	47	1399	96.7%
Ipv6 0 Pad	35	1415	97.9%
Ipv6 8 Pad	43	1403	97.0%

An IPv6/TCP Endpoint packet on a 1500 byte Ethernet has:

- 14 Byte Ethernet Header
- 40 Byte IPv6 Header
- 20 Byte TCP header

Leaving 1426 bytes of payload.

When adding ZDP over IP Substrate headers, the available payload is:

ZDP Encapsulation	Overhead	Payload	Percent of NonZPR
IPv4 0 Pad	39	1387	97.2%
IPv4 8 Pad	47	1379	96.7%
Ipv6 0 Pad	35	1395	97.8%
Ipv6 8 Pad	43	1383	97.0%

Thus, we should see a reduction of about 3% when running ZPR over IP/UDP over Ethernet.

TODO: What is the difference between these tables?

6.1.5. Traffic Latency, Visa Acquisition, and Heralding

Traffic for a connection cannot flow through the ZPR network until a Visa has been obtained and then installed in the network. In addition, obtaining the Visa may involve Endpoint authentication steps.

Latency due to Visa acquisition is beyond the scope of this note. We could have mitigated by prospectively forwarding packets to the destination Endpoint on the assumption that a Visa would eventually be acquired. Such traffic would be held up (perhaps at the Egress Dock) until the Visa is granted and then delivered. There are a number of technical issues with doing this (such as buffer space at the Egress Dock, Endpoint-level TCP timeouts and retransmissions, and so on). However, the biggest argument against it is that it allows traffic for which no Visa has (yet) been acquired to traverse the network, violating one of the core principles of ZDP, and perhaps opening attack vectors.

Once a Visa has been acquired, the process for installing the visa in the network runs in parallel with, and slightly ahead of, the first Endpoint Packets transiting the network. The heralding process is, briefly (assuming success in all stages),

1) Forwarder *A* sends a Visa to Forwarder *B*.
2) *B* sends the Visa on to Forwarder *C*. *B* also sends a response, indicating that it accepts the Visa, back to *A*.
3) *A* sends the first packet to *B*. *C* sends a response, indicating acceptance of the Visa, back to *B*.
4) B sends the first packet to *C*.
5) ...

As can be seen, the first packet is delayed by approximately 1 Forwarder-to-Forwarder round-trip time, as opposed to, say, an Ingress-Dock to Egress-Dock round-trip-time.

Note that if the Visa cannot be installed end-to-end, with the *n*th forwarder in the path rejecting the Visa, the first packet will have transited the network to the *n-1*th forwarder, where it will be discarded. This is no worse than normal IP behavior,

The delay could be mitigated by piggybacking the first packet with the Visa herald or sending the first packet immediately after sending the Visa herald, as opposed to waiting for a response. However, the latency does not seem excessive (as opposed to the latency in obtaining the Visa) so since these are somewhat more complex than the method currently specified, we do not believe these mitigations are necessary.

6.2. Scaling

Scaling goals are based on the largest suspected *single network* that would deploy ZPR. Our initial deployment model was that ZPR would be deployed by organizations in order to serve the organization's members and stake holders. For example: a large corporation might deploy one (or more) ZPR networks to support its employees, resources (*e.g.*, servers and databases), and internal devices (*e.g.*, IoT devices, factory and warehouse equipment, or building infrastructure). It might deploy additional ZPR networks in support of its customers, partners, and suppliers. A large national government might do similarly.

The question then is how big?

If we take the US Federal Government as an example, there are between O(2,000,000) and O(5,000,000) employees and contractors. This includes uniformed military. Large corporations such as Amazon and Walmart are similarly sized. In addition, a brief look at the author's computer's network status indicates approximately 100 TCP and UDP connections are typically active. This gives fair starting points to make some working assumptions.

We can assume that each TCP or UDP connection/session indicates a different ZPR Flow. Thus, a network might have O(200m) to O(500m) active Flows in it at any one time. These flows would not all cross a single point in the network (assuming a reasonable network design). However, it is likely that large numbers of flows would implode at certain servers within a network. For example, every machine would probably have an active Flow with the DNS servers and all employees in a company are likely to have Active flows with a mail server, and so on. All of these servers would, presumably, be distributed in some manner (*e.g.* a single MS Exchange server might be able to handle O(1000) to O(10000) users). So it might be reasonable to see O(1000) to O(10000) flows imploding into a single physical server and Dock serving the server.

We must also consider how many flows might traverse a particular Forwarder. This is harder since the design of the network will directly affect how many flows are likely to traverse any given forwarder. The critical scaling resource in a forwarder is the memory used by the lookup process, which scales based on the lookup algorithm employed, and the memory holding the information necessary to forward a packet (such as identification of the output Link, some statistics and rate limiters, and so on). This latter memory scales linearly with the number of flows.

The scaling properties of the Visa ID lookup algorithm will, obviously, depend on the algorithm in use. This is impossible to predict. Since the lookup key, the Visa ID, is assigned by the Forwarder doing the lookup, it is reasonable to assume that the keys are assigned in a manner that is “easy” to look up. Two of the easiest lookups are

1) The key is the direct address, in memory, of the forwarding information. In this case, no additional memory is required for the actual lookup so there is no scaling concern,
2) The key is an index into an array of pointers to the forwarding table. In this case, scaling is O(Number of IDs). So 1M Visa IDs would require 4 or 8 MB of memory, depending on the pointer size. This is also not a significant issue.

If we assume that a single flow requires 256 bytes of data, then 1M flows would be 256M bytes of data for the forwarding information tables. This is large, but not unreasonable for a dedicated software-based forwarder. Hardware forwarders would be more problematic, but a new design, based on the most recent memories and chip technologies, would easily be able to accommodate this. 10 million flows would require 2.56GB, again not impossible. So, we can conclude that scaling, with regard to the forwarding information tables, is a challenge, but not an impossible task.

7. ZDP Keying Protocol

Earlier versions of ZDP proposed IKEv2[1] to provide Dock-to-Adapter and Forwarder-to-Forwarder key management functionality (*e.g.*, exchange of certificates and/or identities, Diffie-Hellman exchange, and so on) for the ZHSA. IKEv2 does not (seem to) rely on having IP underneath it – it operates over UDP so all it really expects is a datagram service. Thus, we believe that it is possible to redefine IKEv2 to operate over non-IP substrate networks.

Other protocols may also be useful.

In earlier work, a place-holder for a notional ZDP Keying Protocol was allocated. This included a basic ZDP packet format.

7.1. ZDP Key Management Packets

ZPR may use IKEv2 for key management between adjacent Forwarders and between an Adapter and its Dock. When ZPR runs over an IP or IP/UDP Substrate, IKEv2 can run in its native mode (*i.e.*, it runs over the Substrate's IP or IP/UDP service as described in the IKEv2 RFC) or as payloads in ZDP Link Management and Docking Session Packets. When ZPR's substrate is not IP (Ethernet, PPP link, raw fiber, etc.) IKEv2 packets are carried in the appropriate ZDP packet directly over the substrate.

> **Ed. Note**: We could define "IKEv2 over Ethernet" and "IKEv2 over PPP" as new standard protocols, however that is not sufficient for our needs. If we run over a raw-fiber link there is no "type" field to separate out the IKEv2 traffic from the other traffic, so we need to provide the demultiplexing.
>
> **Ed. Note**: If it is not practical to recast IKEv2 to run over any old datagram service, or IKEv2 turns out to be otherwise unsuitable, then we'll need to develop our own key-exchange protocol.

The Link Management ZDP Packet's Management Message field carries the complete IKEv2 message:

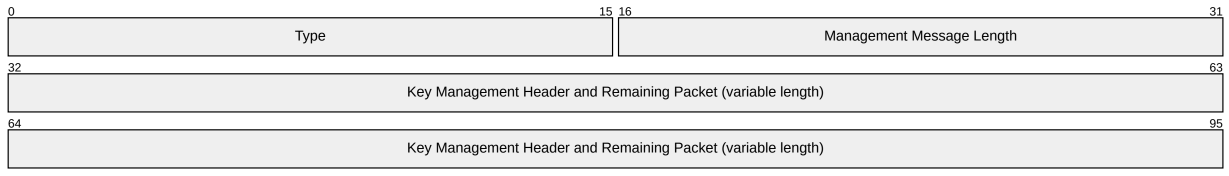

0
15
16
31
Type
Management Message Length
32
63
Key Management Header and Remaining Packet (variable length)
64
95
Key Management Header and Remaining Packet (variable length)

Figure 7.1: Key Management Message Format

Where:

Type Indicates the key management protocol in use. The following values are defined:

[1]Kaufman et al. (2014)

Type	Protocol
0	None
1	IKEv2
2	Noise[2]
255	Experimental

Management Message Length The length of the management message, including the type and length fields (that is, this value must be at least 4).

Key Management Header and Remaining Packet The actual key-management protocol message.

> **Ed. Note**: need to add discussion of how the key protocol is agreed on between peers (*e.g.*, if I speak #1 & #2 and you only speak #1, how so we agree on that? Is it in the config?),

Prior to establishing a SA, or whenever all previously established SAs have terminated, Key Management messages MUST be sent with ZPI 0 (which selects a hard coded SA that does no encryption and just generates an Internet-standard 16-bit checksum for error detection).

This work was deferred. Instead, the key management is done by the trusted management services.

> **Ed. Note**: If we cannot redefine either ESP or IKEv2 as needed then we shall either A) need to find an alternative or B) define ZPR-specific protocols for these tasks.

> **Ed. Note**: If we can redefine ESP and IKEv2 as described this may represent an interesting opportunity to make a contribution to the IETF (establishing AI in the community, etc., etc.). If/when we bring ZPR to the IETF then, we would have some street cred.

[2]Noise Key Management Protocol, http://www.noiseprotocol.org

8. ZPR ARP Protocol

Earlier versions of ZDP included an ARP-like protocol to be used to perform address resolution when ZDP runs directly over Ethernet.

ZPR ARP is a non-flow oriented management message.

8.1. Message Format

The ZPR ARP Management Message Data has the following format:

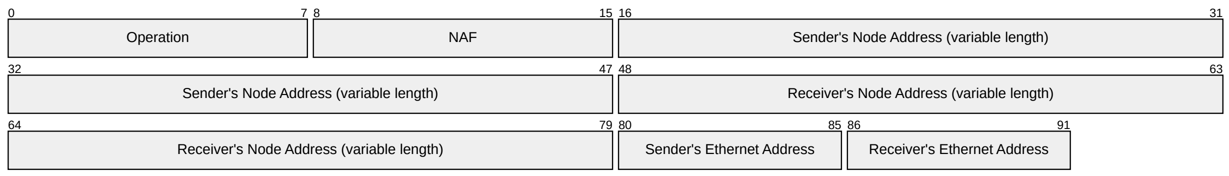

0
7
8
15
16
31
Operation
NAF
Sender's Node Address (variable length)
32
47
48
63
Sender's Node Address (variable length)
Receiver's Node Address (variable length)
64
79
80
85
86
91
Receiver's Node Address (variable length)
Sender's Ethernet Address
Receiver's Ethernet Address

Figure 8.1: ZPR Arp Management Data Format

Where:

OP Is an operation code. It indicates the operation being performed:

Value	Operation
0	Noop
1	Request
2	Response
3-255	Unassigned

Table 8.1: ZPR Arp Operation Codes

NAF Is the address family of the Node Addresses. This field indicates the address family from which the Node Addresses are derived as well as their length. Address Families are:

Value	Address Family	Address Length
0	Unassigned	NA
1	IPv4	4 bytes
2	IPv6	16 bytes
3-255	Unassigned	NA

Table 8.2: ZPR Arp Address Familiies

Senders Node Address Is the ZPR Node Address of the sender. The length of this field depends on the address family.

Receivers Node Address Is the ZPR Node Address of the receiver. The length of this field depends on the address family.
Sender's Ethernet Address The Ethernet address of the node sending the ZARP packet.
Receivers Ethernet Address The Ethernet Address of the recipient of the packet.

8.2. Procedure

ZPR ARP is used when

1) A Node needs to establish a Docking Session or Link with another Node,
2) The Nodes are on the same Ethernet,
3) The Nodes run ZDP directly over raw Ethernet, and
4) The Nodes know the ZPR Node Address of the peer (typically an IP address) but do not know the Ethernet MAC Address of the peer.

ZPR ARP is used to determine the binding between a ZPR Node Address and that Node's Ethernet MAC Address.

ZPR ARP messages are carried as Management Message Data in ZDP packets (see § 4.2.7).

ZPR ARP messages may be exchanged before security associations are established between the peers. If so, the ZPI identifies just the ZPR Configuration in use. The NULL security association is used (see § 3.24.2). The ZPR ARP messages are transmitted in the clear and are presumed to not be modified in transit. This does not represent a significant security hole since immediately following the ZPR ARP exchange, the peers will bring up the Link or Docking Session and perform the necessary cryptographic authentication with each other.

ZPR ARP may be used only when the Substrate is raw Ethernet. If a ZPR ARP message is received when operating over any other Substrate the message MUST be silently discarded, and the Link or Docking Session SHOULD be terminated.

ZPR ARP is used over a Link between adjacent Forwarders or over a Docking Session between a Dock and Adapter when the Link or Docking Session is operating directly over an Ethernet Substrate. When a Forwarder, Dock, or Adapter wishes to send a ZDP packet to a peer it performs the following steps:

1) It checks to see if it has a mapping from the peer's ZPR Node Address to the peer's Ethernet Substrate address. If it does, it creates an Ethernet packet whose destination address is the peer's Ethernet address and transmits the packet. No further processing with regard to ZPR ARP is performed.
2) If it does not have a mapping it creates and sends a ZPR ARP request:
    a. The destination Ethernet address is the Ethernet broadcast address (ff-ff-ff-ff-ff-ff).
    b. The Ethernet source address is the Ethernet address of the interface via which the sender will transmit the packet.
    c. The Ethernet type is ZDP.
    d. ZDP ARP Operation is REQUEST.
    e. Address Family is the address family of the Node Addresses.

   f. Sender's Node Address is the Node Address of the node sending the packet.
   g. Receivers Node Address is the Node Address for which the sender is attempting to determine an Ethernet Address.

3) It sends the ZPR ARP Request packet. It sets a timer for 1 second and a retransmission counter to 3 and then performs other tasks, or sleeps. . .).
4) When the timer expires, it decrements the transmission counter. If the decremented counter is 0 then the transmission of the original packet is deemed to have failed; any queued-up packets waiting for the binding are silently discarded and failure actions are taken. If the counter is not 0, another request is generated and sent. The timer is restarted. The transmission counter is NOT reset. The node then performs other tasks.

When a Node (either the requesting Node or responding Node) receives a ZPR ARP packet it performs the following steps.

1) It validates the packet. If the packet is not deemed valid it is silently discarded. Validity checks are:
   a. The destination MAC address is either the MAC address of the Ethernet interface via which the packet was received, or it is the Ethernet broadcast address: ff-ff-ff-ff-ff-ff.
   b. The source address field of the Ethernet header MUST NOT be an Ethernet group address (multicast or broadcast).
   c. The Ethernet Type is as defined in the Connection Policies for ZDP.
   d. ZDP Arp Operation is either Request or Response (0x01 or 0x02, respectively). If the message is a Response, then the destination MAC address in the Ethernet Header MUST be one of the receiving node's Ethernet addresses.
   e. The Receiver NODE Address must be one of the receiver's Node Addresses.
2) The receiver makes a binding between the sender's Node Address and sender's MAC address.
3) If the message is a Request, a Response is generated
   a. The operation is changed from Request to Response.
   b. The sender and receiver Node Address fields are swapped.
   c. The destination MAC address of the response is set in the source MAC address of the request.
   d. The source MAC address of the request is set to the MAC address of the Ethernet interface via which the response will be transmitted.
   e. The packet is transmitted.
4) If the message is a response, then
   a. If any timers are running, waiting for the response, they are cancelled.
5) Any packets that are queued up waiting for the binding are then transmitted.

Other Points:

1) Failure Actions
   If a ZPR ARP request fails (that is, several requests are sent but no response is received) it is declared a "Failure". Besides reporting the failure, the Connection Policy on the Node initiating the ZPR ARP operation declares how to handle the failure. Options include actions such as A) Terminating the attempt to bring up the Docking Session or Link, B) Waiting for some amount of time and then reattempting, or C) Waiting for some other stimulus and then reattempting. The Policy also controls when and how the failures are reported.

2) Waiting Packets
   ZPR ARP Requests are generated because there is a packet to be transmitted that requires a binding that is not available on the transmitting node. The question is what to do with the packet (and subsequent ones that need the same binding).

ZPR ARP recommends that either the most recent packet that needs the binding or N most recent packets (where N is a very small number, less than 10) be kept while waiting for a binding to be completed. The reasoning is:

a. Saving the earlier packets (that is, discarding ensuing packets) seems bad in that the following packets may represent newer or more current information on the part of the overlying flow.

b. Saving many packets (vs saving 1, or a small, bounded, number) could be a large resource drain.

c. Saving no packets seems like gratuitous inefficiency.

d. IETF Host Requirements, RFC1122[1][2] recommends that at least one (the latest) packet SHOULD be queued. RFC1122 also presents some useful rationale as to why this is the preferred mode.

3) Binding Timeouts
   Node/Substrate address bindings should time out. We suggest that bindings have a maximum lifetime of 5 minutes and SHOULD be 1 minute. See section 2.3.2 of RFC1122.

4) See section 2.3 of RFC1122 for more guidance.

See § 5.3 for a discussion on why ARP[3] or IPv6 Neighbor Discovery[4] are not used instead.

Braden, Robert T. 1989. "Requirements for Internet Hosts - Communication Layers." Request for Comments. RFC 1122; RFC Editor. https://doi.org/10.17487/RFC1122.

Bradner, Scott O. 1997. "Key Words for Use in RFCs to Indicate Requirement Levels." Request for Comments. RFC 2119; RFC Editor. https://doi.org/10.17487/RFC2119.

Douglas, Dave, and F. Kastenholz. 2020. "ZPR Terminology." 4. Zero-Trust Packet Routing. https://github.com/org-zpr/zpr-rfcs/blob/main/pdf/4-Terminology.pdf.

Droms, Ralph. 1997. "Dynamic Host Configuration Protocol." Request for Comments. RFC 2131; RFC Editor. https://doi.org/10.17487/RFC2131.

Floyd, S., and V. Jacobson. 1993. "Random Early Detection Gateways for Congestion Avoidance." *IEEE/ACM Transactions on Networking* 1 (4): 397–413. https://doi.org/10.1109/90.251892.

[1]Braden (1989)

[2]RFC1122, while old, has a great deal of valid and compelling reasoning.

[3]Plummer (1982)

[4]Simpson et al. (2007)

Kaufman, Charlie, Paul E. Hoffman, Yoav Nir, Pasi Eronen, and Tero Kivinen. 2014. “Internet Key Exchange Protocol Version 2 (IKEv2).” Request for Comments. RFC 7296; RFC Editor. https://doi.org/10.17487/RFC7296.

Kent, Stephen. 2005. “IP Encapsulating Security Payload (ESP).” Request for Comments. RFC 4303; RFC Editor. https://doi.org/10.17487/RFC4303.

Nir, Yoav, and Valery Smyslov. 2016. “Protecting Internet Key Exchange Protocol Version 2 (IKEv2) Implementations from Distributed Denial-of-Service Attacks.” Request for Comments. RFC 8019; RFC Editor. https://doi.org/10.17487/RFC8019.

Plummer, David C. 1982. “An Ethernet Address Resolution Protocol: Or Converting Network Protocol Addresses to 48.bit Ethernet Address for Transmission on Ethernet Hardware.” Request for Comments. RFC 826; RFC Editor. https://doi.org/10.17487/RFC0826.

Simpson, William A. 1994. “The Point-to-Point Protocol (PPP).” Request for Comments. RFC 1661; RFC Editor. https://doi.org/10.17487/RFC1661.

Simpson, William A., Dr. Thomas Narten, Erik Nordmark, and Hesham Soliman. 2007. “Neighbor Discovery for IP Version 6 (IPv6).” Request for Comments. RFC 4861; RFC Editor. https://doi.org/10.17487/RFC4861.