Network Working Group
Request for Comments: 5352
Category: Experimental
R. Stewart
Q. Xie
The Resource Group
M. Stillman
Nokia
M. Tuexen
Muenster Univ. of Applied Sciences
September 2008

Aggregate Server Access Protocol (ASAP)

Status of This Memo

This memo defines an Experimental Protocol for the Internet community. It does not specify an Internet standard of any kind. Discussion and suggestions for improvement are requested. Distribution of this memo is unlimited.

Abstract

Aggregate Server Access Protocol (ASAP; RFC 5352), in conjunction with the Endpoint Handlespace Redundancy Protocol (ENRP; RFC 5353), provides a high-availability data transfer mechanism over IP networks. ASAP uses a handle-based addressing model that isolates a logical communication endpoint from its IP address(es), thus effectively eliminating the binding between the communication endpoint and its physical IP address(es), which normally constitutes a single point of failure.

In addition, ASAP defines each logical communication destination as a pool, providing full transparent support for server pooling and load sharing. It also allows dynamic system scalability -- members of a server pool can be added or removed at any time without interrupting the service.

ASAP is designed to take full advantage of the network level redundancy provided by the Stream Transmission Control Protocol (SCTP; RFC 4960). Each transport protocol, other than SCTP, MUST have an accompanying transport mapping document. It should be noted that ASAP messages passed between Pool Elements (PEs) and ENRP servers MUST use the SCTP transport protocol.

The high-availability server pooling is gained by combining two protocols, namely ASAP and ENRP, in which ASAP provides the user interface for Pool Handle to address translation, load sharing management, and fault management, while ENRP defines the high-availability Pool Handle translation service.

Table of Contents

1. Introduction

The Aggregate Server Access Protocol (ASAP), when used in conjunction with Endpoint Name Resolution Protocol [RFC5353], provides a high-availability data-transfer mechanism over IP networks. ASAP uses a handle-based addressing model that isolates a logical communication endpoint from its IP address(es), thus effectively eliminating the binding between the communication endpoint and its physical IP address(es), which normally constitutes a single point of failure.

When multiple receiver instances exist under the same handle (aka a server pool), an ASAP Endpoint will select one Pool Element (PE), based on the current load sharing policy indicated by the server pool, and deliver its message to the selected PE.

While delivering the message, ASAP can be used to monitor the reachability of the selected PE. If it is found unreachable, before notifying the message sender (an ASAP User) of the failure, ASAP can automatically select another PE (if one exists) under that pool and attempt to deliver the message to that PE. In other words, ASAP is capable of transparent failover amongst PE instances within a server pool.

ASAP depends on ENRP, which provides a high-availability Pool Handlespace. ASAP is responsible for the abstraction of the underlying transport technologies, load distribution management, fault management, as well as presentation to the upper layer (aka an ASAP User) via a unified primitive interface.

When SCTP [RFC4960] is used as the transport layer protocol, ASAP can seamlessly incorporate the link-layer redundancy provided by SCTP.

This document defines the ASAP portion of the high-availability server pool.

1.1. Definitions

This document uses the following terms:

ASAP User: Either a PE or Pool User (PU) that uses ASAP.

Business Card: When presented by a PU or PE, it specifies the pool the sender belongs to and provides a list of alternate PEs in case of failovers.

Operational Scope: The part of the network visible to pool users by a specific instance of the reliable server pooling protocols.

Pool (or Server Pool): A collection of servers providing the same application functionality.

Pool Handle: A logical pointer to a pool. Each server pool will be identifiable in the operational scope of the system by a unique Pool Handle.

Pool Element: A server entity having registered to a pool.

Pool User: A server pool user.

Pool Element Handle (or Endpoint Handle): A logical pointer to a particular Pool Element in a pool, consisting of the Pool Handle and a destination transport address of the Pool Element.

Handlespace: A cohesive structure of Pool Handles and relations that may be queried by an internal or external agent.

Home ENRP Server: The ENRP server to which a PE or PU currently sends all namespace service requests. A PE must only have one Home ENRP server at any given time, and both the PE and its Home ENRP server MUST know and keep track of this relationship. A PU should select one of the available ENRP servers as its Home ENRP server, but the collective ENRP servers may change this by the sending of an ASAP_ENDPOINT_KEEP_ALIVE message.

ENRP Client Channel: The communication channel through which an ASAP User sends all namespace service requests. The client channel is usually defined by the transport address of the Home ENRP server and a well-known port number. The channel MAY make use of multicast or a named list of ENRP servers.

Network Byte Order: Most significant byte first, aka Big Endian.

Transport Address: A transport address is traditionally defined by Network Layer address, Transport Layer protocol and Transport Layer port number. In the case of SCTP running over IP, a transport address is defined by the combination of an IP address and an SCTP port number (where SCTP is the Transport protocol).

1.2. Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in [RFC2119].

1.3. Organization of This Document

Section 2 details the ASAP message formats. In Section 3, we provide detailed ASAP procedures for the ASAP implementer. Section 4 summarizes which messages need to be supported by which nodes, and Section 5 describes the usage of SCTP. In Section 6, details of the ASAP interface are given, focusing on the communication primitives between ASAP, the applications above ASAP, and ASAP itself, and the communications primitives between ASAP and SCTP (or other transport layers). Also included in this discussion are relevant timers and configurable parameters, as appropriate. Section 7 provides threshold and protocol variables.

It should be noted that variables, timers, and constants are used in the text when necessary. The complete list can be found in Section 7.

1.4. Scope of ASAP

The requirements for high availability and scalability do not imply requirements on shared state and data. ASAP does not provide transaction failover. If a host or application fails during the processing of a transaction, this transaction may be lost. Some services MAY provide a way to handle the failure, but this is not guaranteed. ASAP MAY provide hooks to assist an application in building a mechanism to share state but ASAP in itself does NOT share any state.

1.4.1. Extent of the Handlespace

The scope of ASAP/ENRP is NOT Internet-wide. The handlespace is neither hierarchical nor arbitrarily large like DNS. A flat peer-to-peer model is detailed. Pools of servers will exist in different administrative domains. For example, suppose the use of ASAP and ENRP is wanted. First, the PU may use DNS to contact an ENRP server. Suppose a PU in North America wishes to contact a server pool in Japan instead of North America. The PU would use DNS to get the list of IP addresses of the Japanese server pool; that is, the ENRP client channel in Japan. From there, the PU would query the Home ENRP server it established and then directly contact the PE(s) of interest.

2. Message Definitions

All messages, as well as their fields described below, shall be in network byte order during transmission. For fields with a length bigger than 4 bytes, a number in a pair of parentheses may follow the field name to indicate the length of the field in number of bytes.

2.1. ASAP Parameter Formats

The basic message format and all parameter formats can be found in [RFC5354]. Note also that *all* ASAP messages exchanged between an ENRP server and a PE MUST use SCTP as transport, while ASAP messages exchanged between an ENRP server and a PU MUST use either SCTP or TCP as transport. PE to PU data traffic MAY use any transport protocol specified by the PE during registration.

2.2. ASAP Messages

This section details the individual messages used by ASAP. These messages are composed of a standard message format found in Section 4 of [RFC5354]. The parameter descriptions can be found in [RFC5354].

The following ASAP message types are defined in this section:

Type	Message Name
0x00	- (Reserved by IETF)
0x01	- ASAP_REGISTRATION
0x02	- ASAP_DEREGISTRATION
0x03	- ASAP_REGISTRATION_RESPONSE
0x04	- ASAP_DEREGISTRATION_RESPONSE
0x05	- ASAP_HANDLE_RESOLUTION
0x06	- ASAP_HANDLE_RESOLUTION_RESPONSE
0x07	- ASAP_ENDPOINT_KEEP_ALIVE
0x08	- ASAP_ENDPOINT_KEEP_ALIVE_ACK
0x09	- ASAP_ENDPOINT_UNREACHABLE
0x0a	- ASAP_SERVER_ANNOUNCE
0x0b	- ASAP_COOKIE
0x0c	- ASAP_COOKIE_ECHO
0x0d	- ASAP_BUSINESS_CARD
0x0e	- ASAP_ERROR
others	- (Reserved by IETF)

Figure 1

2.2.1. ASAP_REGISTRATION Message

The ASAP_REGISTRATION message is sent by a PE to its Home ENRP server to either create a new pool or to add itself to an existing pool. The PE sending the ASAP_REGISTRATION message MUST fill in the Pool Handle parameter and the Pool Element parameter. The Pool Handle parameter specifies the name to be registered. The Pool Element parameter MUST be filled in by the registrant, as outlined in Section 3.1. Note that the PE sending the registration message MUST send the message using an SCTP association. Furthermore, the IP address(es) of the PE that is registered within the Pool Element parameter MUST be a subset of the IP address(es) used in the SCTP association, regardless of the registered transport protocol.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x01 |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Element Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pool Handle Parameter:

See [RFC5354].

Pool Element Parameter:

See [RFC5354].

2.2.2. ASAP_DEREGISTRATION Message

The ASAP_DEREGISTRATION message is sent by a PE to its Home ENRP server to remove itself from a pool to which it registered.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x02 |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                    PE Identifier Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+++
```

Pool Handle Parameter:

See [RFC5354].

PE Identifier Parameter:

See [RFC5354].

The PE sending the ASAP_DEREGISTRATION MUST fill in the Pool Handle and the PE identifier parameter in order to allow the ENRP server to verify the identity of the endpoint. Note that de-registration is NOT allowed by proxy; in other words, a PE may only de-register itself.

2.2.3. ASAP_REGISTRATION_RESPONSE Message

The ASAP_REGISTRATION_RESPONSE message is sent in response by the Home ENRP server to the PE that sent an ASAP_REGISTRATION message.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x03 |0|0|0|0|0|0|0|R|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                    PE Identifier Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                   Operational Error (optional)                :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

R (Reject) Flag:

When set to '1', this flag indicates that the ENRP server sending this message has rejected the registration. Otherwise, when this flag is set to '0', this indicates the registration has been granted.

Pool Handle Parameter:

See [RFC5354].

PE Identifier Parameter:

See [RFC5354].

Operational Error Parameter (optional):

See [RFC5354].

This parameter is included if an error or some atypical events occurred during the registration process. When the R flag is set to '1', this parameter, if present, indicates the cause of the rejection. When the R flag is set to '0', this parameter, if present, serves as a warning to the registering PE, informing it that some of its registration values may have been modified by the ENRP server. If the registration was successful and there is no warning, this parameter is not included.

2.2.4. ASAP_DEREGISTRATION_RESPONSE Message

The ASAP_DEREGISTRATION_RESPONSE message is returned by the Home ENRP server to a PE in response to an ASAP_DEREGISTRATION message or due to the expiration of the registration life of the PE in the pool.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x04 |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                    PE Identifier Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                  Operational Error (optional)                 :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pool Handle Parameter:

See [RFC5354].

PE Identifier Parameter:

See [RFC5354].

Operational Error:

See [RFC5354].

This parameter is included if an error or some atypical events occurred during the de-registration process. If the de-registration was successful this parameter is not included.

2.2.5. ASAP_HANDLE_RESOLUTION Message

The ASAP_HANDLE_RESOLUTION message is sent by either a PE or PU to its Home ENRP server to resolve a Pool Handle into a list of Pool Elements that are members of the pool indicated by the Pool Handle.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x05 |0|0|0|0|0|0|0|S|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                      Pool Handle Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

The 'S' bit:

The 'S' bit, if set to '1', requests the Home ENRP server to send updates to this Pool dynamically when the Pool changes for the lifetime of the SCTP association. Dynamic updates to the pool will consist of additional ASAP_HANDLE_RESOLUTION_RESPONSE messages, without the user needing to send in an ASAP_HANDLE_RESOLUTION.

If the 'S' bit is set to '0', no Dynamic updates are requested.

Note that if a new Home ENRP server is adopted, any 'dynamic update request' will need to be re-sent to the new Home ENPR server if the endpoint would like to continue to receive updates. In other words, the ENRP servers do NOT share state regarding which of its PU's are requesting automatic update of state. Thus, upon change of Home ENRP server, the PU will need to re-send an ASAP_HANDLE_RESOLUTION message with the 'S' bit set to '1'. Note also, that the 'S' bit will only cause Dynamic update of a Pool when the Pool exists. If a negative response is returned, no further updates to the Pool (when it is created) will occur.

Pool Handle Parameter:

See [RFC5354].

2.2.6. ASAP_HANDLE_RESOLUTION_RESPONSE Message

The ASAP_HANDLE_RESOLUTION_RESPONSE message is sent in response by the Home ENRP server of the PU or PE that sent an ASAP_HANDLE_RESOLUTION message or is sent periodically upon Pool changes if the PU has requested Dynamic updates.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Type = 0x06 |0|0|0|0|0|0|0|A|        Message Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                     Pool Handle Parameter                     :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :             Overall PE Selection Policy (optional)            :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                Pool Element Parameter 1 (optional)            :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                              ...                              :
 :                                                               :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                Pool Element Parameter N (optional)            :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                    Operational Error (optional)               :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

'A' bit:

This bit is set to '1' if the ENRP server accepts the request to send automatic updates (i.e., the 'S' bit was set on the request). If this bit is set to '0', either the ENRP server does NOT support automatic updates, it has resource issues and cannot supply this feature, or the user did not request it.

Pool Handle Parameter:

See [RFC5354].

Overall PE Selection Policy (optional):

See [RFC5354].

This parameter can be present when the response is positive. If present, it indicates the overall pool member selection policy of the pool. If not present, a Round-Robin overall pool member selection policy is assumed. This parameter is not present when the response is negative.

Note, any load policy parameter within a Pool Element parameter (if present) MUST be ignored, and MUST NOT be used to determine the overall pool member selection policy.

Pool Element Parameters (optional):

See [RFC5354].

When the response is positive, an array of PE parameters are included, indicating the current information about the PEs in the named pool. At least one PE parameter MUST be present. When the response is negative, no PE parameters are included.

Operational Error (optional):

See [RFC5354].

The presence of this parameter indicates that the response is negative (the handle resolution request was rejected by the ENRP server). The cause code in this parameter (if present) will indicate the reason the handle resolution request was rejected (e.g., the requested Pool Handle was not found). The absence of this parameter indicates that the response is positive.

2.2.7. ASAP_ENDPOINT_KEEP_ALIVE Message

The ASAP_ENDPOINT_KEEP_ALIVE message is sent by an ENRP server to a PE. The ASAP_ENDPOINT_KEEP_ALIVE message is used to verify that the PE is reachable and requires the PE to adopt the sending server as its new Home ENRP server if the 'H' bit is set to '1'. Regardless of the setting of the 'H' bit, an ASAP Endpoint MUST respond with an ASAP_ENDPOINT_KEEP_ALIVE_ACK to any ASAP_ENDPOINT_KEEP_ALIVE messages that arrive.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x07 |0|0|0|0|0|0|0|H|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Server Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

H (Home ENRP server) Flag:

When set to '1', indicates that the ENRP server that sends this message wants to be the Home ENRP server of the receiver of this message.

Server Identifier: 32 bits (unsigned integer)

This is the ID of the ENRP server, as discussed in [RFC5353].

Pool Handle Parameter:

See [RFC5354].

2.2.8. ASAP_ENDPOINT_KEEP_ALIVE_ACK Message

The ASAP_ENDPOINT_KEEP_ALIVE_ACK message is sent by a PE in response to an ASAP_ENDPOINT_KEEP_ALIVE message sent by an ENRP server.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x08 |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                    PE Identifier Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pool Handle Parameter:

See [RFC5354].

PE Identifier Parameter:

See [RFC5354].

2.2.9. ASAP_ENDPOINT_UNREACHABLE Message

The ASAP_ENDPOINT_UNREACHABLE message is sent by either a PE or PU to its Home ENRP server to report an unreachable PE.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x09 |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                     Pool Handle Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                    PE Identifier Parameter                    :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pool Handle Parameter:

See [RFC5354].

PE Identifier Parameter:

See [RFC5354].

2.2.10. ASAP_SERVER_ANNOUNCE Message

The ASAP_SERVER_ANNOUNCE message is sent by an ENRP server such that PUs and PEs know the transport information necessary to connect to the ENRP server.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x0a |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                       Server Identifier                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                       Transport Param #1                      :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                       Transport Param #2                      :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                                                               :
:                             .....                             :
:                                                               :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                       Transport Param #n                      :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Server Identifier: 32 bits (unsigned integer)

This is the ID of the ENRP server, as discussed in [RFC5353].

Transport Parameters (optional):

See [RFC5354] for the SCTP and TCP Transport parameters.

Only SCTP and TCP Transport parameters are allowed for use within the SERVER_ANNOUNCE message.

2.2.11. ASAP_COOKIE Message

The ASAP_COOKIE message is sent by a PE to a PU, allowing the PE to convey information it wishes to share using a control channel.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x0b |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                          Cookie Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cookie Parameter :

See [RFC5354].

2.2.12. ASAP_COOKIE_ECHO Message

The ASAP_COOKIE_ECHO message is sent by a PU to a new PE when it detects a failure with the current PE to aid in failover. The Cookie Parameter sent by the PE is the latest one received from the failed PE.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Type = 0x0c |0|0|0|0|0|0|0|0|        Message Length         |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
:                          Cookie Parameter                     :
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Cookie Parameter:

See [RFC5354].

2.2.13. ASAP_BUSINESS_CARD Message

The ASAP_BUSINESS_CARD message is sent by a PU to a PE or from a PE to a PU using a control channel to convey the pool handle and a preferred failover ordering.

```
  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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Type = 0x0d |0|0|0|0|0|0|0|0|        Message Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                      Pool Handle Parameter                    :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                    Pool Element Parameter-1                   :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                               ..                              :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                    Pool Element Parameter-N                   :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Pool Handle Parameter:

See [RFC5354].

Pool Element Parameters:

See [RFC5354].

2.2.14. ASAP_ERROR Message

The ASAP_ERROR message is sent in response by an ASAP Endpoint receiving an unknown message or an unknown parameter to the sending ASAP Endpoint to report the problem or issue.

```
 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
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 |   Type = 0x0e |0|0|0|0|0|0|0|0|        Message Length         |
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
 :                  Operational Error Parameter                  :
 +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Operational Error Parameter:

See [RFC5354].

When an ASAP Endpoint receives an ASAP message with an unknown message type or a message of known type that contains an unknown parameter, it SHOULD handle the unknown message or the unknown parameter according to the unrecognized message and parameter handling rules, defined in Section 3.

According to the rules, if an error report to the message sender is needed, the ASAP endpoint that discovered the error SHOULD send back an ASAP_ERROR message that includes an Operational Error parameter with the proper cause code, cause length, and case-specific information.

3. Procedures

This section will focus on the methods and procedures used by an internal ASAP Endpoint. Appropriate timers and recovery actions for failure detection and management are also discussed. Also, please note that ASAP messages sent between a PE and PU are identified by an SCTP Payload Protocol Identifier (PPID).

3.1. Registration

When a PE wishes to initiate or join a server pool, it MUST use the procedures outlined in this section for registration. Often, the registration will be triggered by a user request primitive (discussed in Section 6.1). The PE MUST register using an SCTP association established between itself and the Home ENRP server. If the PE has not established its Home ENRP server, it MUST follow the procedures specified in Section 3.6.

Once the PE's ASAP Endpoint has established its Home ENRP server, the following procedures MUST be followed to register:

R1) The PE's SCTP endpoint used to communicate with the Home ENRP server MUST be bound to all IP addresses that will be used by the PE (regardless of which transport protocol will be used to service user requests to the PE).

R2) The PE's ASAP Endpoint MUST formulate an ASAP_REGISTRATION message, as defined in Section 2.2.1. In formulating the message, the PE MUST:

  R2.1) Fill in the Pool Handle parameter to specify which server pool the ASAP Endpoint wishes to join.

  R2.2) Fill in the PE identifier using a good-quality randomly generated number ([RFC4086] provides some information on randomness guidelines).

R2.3) Fill in the Registration Lifetime parameter with the number of seconds that this registration is valid for. Note that a PE that wishes to continue service MUST re-register before the registration expires.

R2.4) Fill in a User Transport parameter to specify the type of transport and the data/control channel usage the PE is willing to support. Note, in joining an existing server pool, the PE MUST follow the overall transport type and overall data/control channel usage of the pool. Otherwise, the registration may be rejected by the ENRP server.

R2.5) Fill in the preferred Pool Member Selection Policy parameter.

R3) Send the ASAP_REGISTRATION message to the Home ENRP server using SCTP.

R4) Start a T2-registration timer.

Note: the PE does not need to fill in the optional ASAP transport parameter. The ASAP transport parameter will be filled in and used by the Home ENRP server.

If the T2-registration timer expires before receiving an ASAP_REGISTRATION_RESPONSE message, or a SEND.FAILURE notification is received from the SCTP layer, the PE shall start the Server Hunt procedure (see Section 3.6) in an attempt to get service from a different ENRP server. After establishing a new Home ENRP server, the PE SHOULD restart the registration procedure.

At the reception of the registration response, the PE MUST stop the T2-registration timer. If the response indicates success, the PE is registered and will be considered an available member of the server pool. If the registration response indicates a failure, the PE must either re-attempt registration after correcting the error or return a failure indication to the PE’s upper layer. The PE MUST NOT re-attempt registration without correcting the error condition.

At any time, a registered PE MAY wish to re-register to either update its member selection Policy Value or registration expiration time. When re-registering, the PE MUST use the same PE identifier.

After successful registration, the PE MUST start a T4-reregistration timer. At its expiration, a re-registration SHOULD be made starting at step R1, including (at completion) restarting the T4-reregistration timer.

Note that an implementation SHOULD keep a record of the number of registration (and re-registration) attempts it makes in a local variable that gets set to zero before the initial registration attempt to the Home ENRP server or after a successful re-registration. If repeated registration timeouts or failures occurs and the local count exceeds the Threshold 'MAX-REG-ATTEMPT', the implementation SHOULD report the error to its upper layer and stop attempting registration.

The ENRP server handles the ASAP_REGISTRATION message according to the following rules:

1. If the named pool does not exist in the handlespace, the ENRP server MUST create a new pool with that handle in the handlespace and add the PE to the pool as its first PE.

   When a new pool is created, the overall member selection policy of the pool MUST be set to the policy type indicated by the first PE, the overall pool transport type MUST be set to the transport type indicated by the PE, and the overall pool data/control channel configuration MUST be set to what is indicated in the Transport Use field of the User Transport parameter by the registering PE.

2. If the named pool already exists in the handlespace, but the requesting PE is not currently a member of the pool, the ENRP server will add the PE as a new member to the pool.

   However, before adding the PE to the pool, the server MUST check if the policy type, transport type, and transport usage indicated by the registering PE is consistent with those of the pool. If different, the ENRP server MUST reject the registration.

3. If the named pool already exists in the handlespace *and* the requesting PE is already a member of the pool, the ENRP server SHOULD consider this as a re-registration case. The ENRP server MUST perform the same tests on policy, transport type, and transport use, as described above. If the re-registration is accepted after the test, the ENRP server SHOULD replace the attributes of the existing PE with the information carried in the received ASAP_REGISTRATION message.

4. After accepting the registration, the ENRP server MUST assign itself the owner of this PE. If this is a re-registration, the ENRP server MUST take over ownership of this PE, regardless of whether the PE was previously owned by this server or by another

server. The ENRP server MUST also record the SCTP transport address from which it received the ASAP_REGISTRATION in the ASAP Transport parameter TLV inside the PE parameter of this PE.

5. The ENRP server may reject the registration due to other reasons such as invalid values, lack of resource, authentication failure, etc.

In all above cases, the ENRP server MUST reply to the requesting PE with an ASAP_REGISTRATION_RESPONSE message. If the registration is accepted, the ENRP server MUST set the R flag in the ASAP_REGISTRATION_RESPONSE to '0'. If the registration is rejected, the ENRP server MUST indicate the rejection by setting the R flag in the ASAP_REGISTRATION_RESPONSE to '1'.

If the registration is rejected, the ENRP server SHOULD include the proper error cause(s) in the ASAP_REGISTRATION_RESPONSE message.

If the registration is granted (either a new registration or a re-registration case), the ENRP server MUST assign itself to be the Home ENRP server of the PE, i.e., to "own" the PE.

> Implementation note: For better performance, the ENRP server may find it both efficient and convenient to internally maintain two separate PE lists or tables -- one is for the PEs that are owned by the ENRP server and the other is for all the PEs owned by their peer(s).

Moreover, if the registration is granted, the ENRP server MUST take the handlespace update action to inform its peers about the change just made. If the registration is denied, no message will be sent to its peers.

3.2. De-Registration

In the event a PE wishes to de-register from its server pool (normally, via an upper-layer request, see Section 6.2), it SHOULD use the following procedure. It should be noted that an alternate method of de-registration is to NOT re-register and to allow the registration life of the PE to expire. In this case, an ASAP_DEREGISTRATION_RESPONSE message is sent to the PE's ASAP Endpoint to indicate the removal of the PE from the pool it registered.

When de-registering, the PE SHOULD use the SCTP association that was used for registration with its Home ENRP server. To de-register, the PE's ASAP Endpoint MUST take the following actions:

D1) Fill in the Pool Handle parameter of the ASAP_DEREGISTRATION message (Section 2.2.2) using the same Pool Handle parameter sent during registration.

D2) Fill in the PE Identifier parameter of the ASAP_DEREGISTRATION message. The identifier MUST be the same as used during registration. The use of the same Pool Handle and Pool Identifier parameters used in registration allows the identity of the PE ASAP Endpoint to be verified before de-registration can occur.

D3) Send the ASAP_DEREGISTRATION message to the Home ENRP server using the PE’s SCTP association.

D4) Start a T3-Deregistration timer.

If the T3-Deregistration timer expires before receiving either an ASAP_REGISTRATION_RESPONSE message, or a SEND.FAILURE notification from the PE’s SCTP endpoint, the PE’s ASAP Endpoint shall start the ENRP Server Hunt procedure (see Section 3.6) in an attempt to get service from another ENRP server. After establishing a new Home ENRP server, the ASAP Endpoint SHOULD restart the de-registration procedure.

At the reception of the ASAP_DEREGISTRATION_RESPONSE, the PE’s ASAP endpoint MUST stop the T3-Deregistration timer.

It should be noted that after a successful de-registration, the PE MAY still receive requests for some period of time. The PE MAY wish to remain active and service these requests or to exit and ignore these requests.

Upon receiving the message, the ENRP server SHALL remove the PE from its handlespace. Moreover, if the PE is the last one of the named pool, the ENRP server will remove the pool from the handlespace as well.

If the ENRP server fails to find any record of the PE in its handlespace, it SHOULD consider the de-registration granted and completed, and send an ASAP_DEREGISTRATION_RESPONSE message to the PE.

The ENRP server may reject the de-registration request for various reasons, such as invalid parameters, authentication failure, etc.

In response, the ENRP server MUST send an ASAP_DEREGISTRATION_RESPONSE message to the PE. If the de-registration is rejected, the ENRP server MUST indicate the rejection by including the proper Operational Error parameter.

It should be noted that de-registration does not stop the PE from sending or receiving application messages.

Once the de-registration request is granted *and* the PE removed from its local copy of the handlespace, the ENRP server MUST take the handlespace update action to inform its peers about the change just made. Otherwise, the ENRP server MUST NOT inform its peers.

3.3. Handle Resolution

At any time, a PE or PU may wish to resolve a handle. This usually will occur when an ASAP Endpoint sends a Pool Handle (Section 6.5.1) to its Home ENRP server or requests a cache population (Section 6.3). It may also occur for other reasons (e.g., the internal ASAP PE wishes to know its peers to send a message to all of them). When an ASAP Endpoint (PE or PU) wishes to resolve a pool handle to a list of accessible transport addresses of the member PEs of the pool, it MUST take the following actions:

NR1) Fill in an ASAP_HANDLE_RESOLUTION message (Section 2.2.5) with the Pool Handle to be resolved.

NR2) If the endpoint does not have a Home ENRP server, start the ENRP Server Hunt procedures specified in Section 3.6 to obtain one. Otherwise, proceed to step NR3.

NR3) If a PE, send the ASAP_HANDLE_RESOLUTION message to the Home ENRP server using SCTP; if a PU, send the ASAP_HANDLE_RESOLUTION message to the Home ENRP server using either TCP or SCTP. If sent from a PE, the SCTP association used for registration SHOULD be used.

NR4) Start a T1-ENRPrequest timer.

If the T1-ENRPrequest timer expires before receiving a response message, the ASAP Endpoint SHOULD take the steps described in Section 3.7.2. If a SEND.FAILURE notification is received from the SCTP or TCP layer, the ASAP Endpoint SHOULD start the Server Hunt procedure (see Section 3.6) in an attempt to get service from a different ENRP server. After establishing a new Home ENRP server, the ASAP Endpoint SHOULD restart the handle resolution procedure.

At the reception of the ASAP_HANDLE_RESOLUTION_RESPONSE message, the ASAP Endpoint MUST stop its T1-ENRPrequest timer. After stopping the T1-ENRPrequest timer, the ASAP Endpoint SHOULD process the message as appropriate (e.g., populate a local cache, give the response to the ASAP User, and/or use the response to send the ASAP User's message).

Note that some ASAP Endpoints MAY use a cache to minimize the number of handle resolutions sent. If a cache is used, it SHOULD:

C1) Be consulted before sending a handle resolution.

C2) Have a stale timeout timer associated with each cache entry. If the cache entry is determined to be stale upon a cache hit, a handle resolution message SHOULD be sent so the cache can be updated.

C3) In the case of a stale cache entry, the implementation may, in parallel, update the cache and answer the request, or it may block the user and wait for an updated cache before proceeding with the users request.

C4) If the cache entry is NOT stale, the endpoint SHOULD NOT send a handle resolution request but instead SHOULD use the entry from the cache.

It should be noted that the impact of using a cache depends on the policy and the requirements of the application. For some applications, cache-usage can increase the performance of the system; for some, it can decrease it.

An ENRP server SHOULD be prepared to receive ASAP_HANDLE_RESOLUTION requests from PUs, either over an SCTP association on the well-known SCTP port, or over a TCP connection on the well-known TCP port.

Upon reception of the ASAP_HANDLE_RESOLUTION message, the ENRP server MUST first look up the pool handle in its handlespace. If the pool exists, the Home ENRP server MUST compose and send back an ASAP_HANDLE_RESOLUTION_RESPONSE message to the requesting PU.

In the response message, the ENRP server SHOULD list all the PEs currently registered in this pool, in a list of PE parameters. The ENRP server MUST also include a pool member selection policy parameter to indicate the overall member selection policy for the pool, if the current pool member selection policy is not Round-Robin.

If the named pool does not exist in the handlespace, the ENRP server MUST reject the handle resolution request by responding with an ASAP_HANDLE_RESOLUTION_RESPONSE message carrying an Unknown Pool Handle error.

3.4. Endpoint Keep Alive

The ASAP_ENDPOINT_KEEP_ALIVE message is sent by an ENRP server to a PE in order to verify it is reachable. If the transport level heartbeat mechanism is insufficient, this message can be used in a heartbeat mechanism for the ASAP level whose goal is determining the health status of the ASAP level in a timely fashion. (The transport level heartbeat mechanism may be insufficient due to either the timeouts or the heartbeat interval being set too long, or, that the transport level heartbeat mechanism's coverage is limited only to the transport level at the two ends.) Additionally, the ASAP_ENDPOINT_KEEP_ALIVE message has value in the reliability of fault detection if the SCTP stack is in the kernel. In such a case, while the SCTP-level heartbeat monitors the end-to-end connectivity between the two SCTP stacks, the ASAP-level heartbeat monitors the end-to-end liveliness of the ASAP layer above it.

The use of the ASAP_ENDPOINT_KEEP_ALIVE message (Section 2.2.7) and the ASAP_ENDPOINT_KEEP_ALIVE_ACK (Section 2.2.8) is described below. Upon reception of an ASAP_ENDPOINT_KEEP_ALIVE message, the following actions MUST be taken:

KA1) The PE must verify that the Pool Handle is correct and matches the Pool Handle sent in its earlier ASAP_REGISTRATION message. If the Pool Handle does not match, the PE MUST silently discard the message.

KA2) Send an ASAP_ENDPOINT_KEEP_ALIVE_ACK (Section 2.2.8) as follows:

- KA2.1) Fill in the Pool Handle parameter with the PE's Pool Handle.

- KA2.2) Fill in the PE Identifier parameter using the PE identifier used by this PE for registration.

- KA2.3) Send the ASAP_ENDPOINT_KEEP_ALIVE_ACK message via the appropriate SCTP association for the ENRP server that sent the ASAP_ENDPOINT_KEEP_ALIVE message.

- KA2.4) If the H flag in the received ASAP_ENDPOINT_KEEP_ALIVE message is set, and the Server Identifier in the message is NOT the identity of your Home ENRP server (or it is not set, e.g., you have a no Home ENRP server) adopt the sender of the ASAP_ENDPOINT_KEEP_ALIVE message as the new Home ENRP server.

3.5. Unreachable Endpoints

Occasionally, an ASAP Endpoint may realize a PE is unreachable. This may occur by a specific SCTP error realized by the ASAP endpoint or via an ASAP User report via the Transport.Failure Primitive (Section 6.9.2). In either case, the ASAP Endpoint SHOULD report the unavailability of the PE by sending an ASAP_ENDPOINT_UNREACHABLE message to any ENRP server. Before sending the ASAP_ENDPOINT_UNREACHABLE message, the ASAP Endpoint should fill in the Pool Handle parameter and PE Identifier parameter of the unreachable endpoint. If the sender is a PE, the message MUST be sent via SCTP. It should be noted that an ASAP Endpoint MUST report no more than once each time it encounters such an event. Additionally, when processing a Transport.Failure Primitive (Section 6.9.2), the ASAP Endpoint MUST NOT send an ASAP_ENDPOINT_UNREACHABLE message unless the user has made a previous request to send data to the PE specified by the primitive.

Upon the reception of an ASAP_ENDPOINT_UNREACHABLE message, an ENRP server MUST immediately send a point-to-point ASAP_ENDPOINT_KEEP_ALIVE message to the PE in question (the H flag in the message SHOULD be set to '0', in this case). If this ASAP_ENDPOINT_KEEP_ALIVE fails (e.g., it results in an SCTP SEND.FAILURE notification), the ENRP server MUST consider the PE as truly unreachable and MUST remove the PE from its handlespace.

If the ASAP_ENDPOINT_KEEP_ALIVE message is transmitted successfully to the PE, the ENRP server MUST retain the PE in its handlespace. Moreover, the server SHOULD keep a counter to record how many ASAP_ENDPOINT_UNREACHABLE messages it has received reporting reachability problem relating to this PE. If the counter exceeds the protocol threshold MAX-BAD-PE-REPORT, the ENRP server SHOULD remove the PE from its handlespace.

Optionally, an ENRP server may also periodically send point-to-point ASAP_ENDPOINT_KEEP_ALIVE (with the H flag set to '0') messages to each of the PEs owned by the ENRP server in order to check their reachability status. If the sending of ASAP_ENDPOINT_KEEP_ALIVE to a PE fails, the ENRP server MUST consider the PE as unreachable and MUST remove the PE from its handlespace. Note, if an ENRP server owns a large number of PEs, the implementation should pay attention not to flood the network with bursts of ASAP_ENDPOINT_KEEP_ALIVE messages. Instead, the implementation MUST distribute the ASAP_ENDPOINT_KEEP_ALIVE message traffic over a time period. This can be achieved by varying the time between two ASAP_ENDPOINT_KEEP_ALIVE messages to the same PE randomly by plus/minus 50 percent.

3.6. ENRP Server Hunt Procedures

Each PU and PE manages a list of transport addresses of ENRP servers it knows about.

If multicast capabilities are used within the operational scope, an ENRP server MUST send periodically every (N+1)*T6-Serverannounce an ASAP_SERVER_ANNOUNCE message (Section 2.2.10), which includes all the transport addresses available for ASAP communication on the multicast ENRP client channel, where N is the number of ENRP servers the server has found via receiving ASAP_SERVER_ANNOUNCE messages. This should result in a message rate of approximately 1 ASAP_SERVER_ANNOUNCE per T6-Serverannounce.

If an ASAP_SERVER_ANNOUNCE message is received by a PU or PE, it SHOULD insert all new included transport addresses into its list of ENRP server addresses and start a T7-ENRPoutdate timer for each address. For all already-known, included transport addresses, the T7-ENRPoutdate timer MUST be restarted for each address. If no transport parameters are included in the ASAP_SERVER_ANNOUNCE message, the SCTP transport protocol is assumed to be used and the source IP address and the IANA-registered ASAP port number is used for communication with the ENRP server. If a T7-ENRPoutdate timer for a transport address expires, the corresponding address is deleted from the managed list of transport addresses of the PU or PE.

If multicast capabilities are not used within the operational scope, each PU and PE MUST have a configured list of transport addresses of ENRP servers.

At its startup, or when it fails to communicate with its Home ENRP server (i.e., timed out on an ENRP request), a PE or PU MUST establish a new Home ENRP server (i.e., set up a TCP connection or SCTP association with a different ENRP server).

To establish a Home ENRP server, the following rules MUST be followed:

SH1) The PE or PU SHOULD try to establish an association or connection, with no more than three ENRP servers. An ASAP Endpoint MUST NOT establish more than three associations or connections.

SH2) The ASAP Endpoint shall start a T5-Serverhunt timer.

SH3) If the ASAP Endpoint establishes an association or connection it MUST stop its T5-Serverhunt timer. The ASAP Endpoint SHOULD also reset the T5-Serverhunt timer to its initial value and then proceed to step SH6.

SH4) If an association or connection establishment fails, the ASAP Endpoint SHOULD try to establish an association or connection using a different transport address.

SH5) If the T5-Serverhunt timer expires, the following should be performed:

SH5.1) The ASAP Endpoint MUST double the value of the T5-Serverhunt timer. Note that this doubling is capped at the value RETRAN.max.

SH5.2) The ASAP Endpoint SHOULD stop the establishment of associations and connections with the transport addresses selected in step SH1.

SH5.2) The ASAP Endpoint SHOULD repeat trying to establish an association or connection by proceeding to step SH1. It SHOULD attempt to select a different set of transport addresses with which to connect.

SH6) The PE or PU shall pick one of the ENRP servers with which it was able to establish an association or connection, and send all subsequent ENRP request messages to this new Home ENRP server.

3.7. Handling ASAP Endpoint to ENRP Server Communication Failures

Three types of failure may occur when the ASAP Endpoint at either the PE or PU tries to communicate with an ENRP server:

A) SCTP send failure

B) T1-ENRPrequest timer expiration

C) Registration failure

3.7.1. SCTP Send Failure

This communication failure indicates that the SCTP layer was unable to deliver a message sent to an ENRP server. In other words, the ENRP server is unreachable.

In such a case, the ASAP Endpoint MUST NOT re-send the undeliverable message. Instead, it SHOULD discard the message and start the ENRP Server Hunt procedure as described in Section 3.6. After finding a new Home ENRP server, the ASAP Endpoint should re-send the request.

Note that an ASAP Endpoint MAY also choose to NOT discard the message, but to queue it for retransmission after a new Home ENRP server is found. If an ASAP Endpoint does choose to discard the message, after a new Home ENRP server is found, the ASAP Endpoint MUST be capable of reconstructing the original request.

3.7.2. T1-ENRPrequest Timer Expiration

When the T1-ENRPrequest timer expires, the ASAP Endpoint should re-send the original request to the ENRP server and restart the T1-ENRPrequest timer. In parallel, the ASAP Endpoint should begin the ENRP server hunt procedures described in Section 3.6.

This should be repeated up to MAX-REQUEST-RETRANSMIT times. After that, an Error.Report notification should be generated to inform the ASAP User, and the ENRP request message associated with the T1-ENRPrequest timer should be discarded. It should be noted that if an alternate ENRP server responds, the ASAP Endpoint SHOULD adopt the responding ENRP server as its new Home ENRP server and re-send the request to the new Home ENRP server.

3.7.3. Registration Failure

Registration failure is discussed in Section 3.1.

3.8. Cookie Handling Procedures

Whenever a PE wants, and a control channel exists, it can send an ASAP_COOKIE message to a PU via the control channel. The PU's ASAP endpoint stores the Cookie parameter and discards an older cookie if it is previously stored.

Note: A control channel is a communication channel between a PU and PE that does not carry data passed to the user. This is accomplished with SCTP by using a PPID to separate the ASAP messages (Cookie and Business Card) from normal data messages.

If the PU's ASAP Endpoint detects a failure and initiates a failover to a different PE, it SHOULD send the latest received cookie parameter in an ASAP_COOKIE_ECHO message to the new PE as the first message on the control channel. Upper layers may be involved in the failover procedure.

The cookie handling procedure can be used for state sharing. Therefore, a cookie should be signed by the sending PE ASAP Endpoint and the cookie should be verified by the receiving PE's ASAP Endpoint. The details of the verification procedure are out of scope for this document. It is only important that the PU always stores the last received Cookie parameter and sends that back unmodified in case of a PE failure.

3.9. Business Card Handling Procedures

When communication begins between a PU and a PE, either of which could be part of a PU/PE combination (i.e., a message is sent between the entities), a PE should always send an ASAP_BUSINESS_CARD message to a PU. A PU should send an ASAP_BUSINESS_CARD message to a PE only if it is part of a PU/PE combination. An ASAP_BUSINESS_CARD message MUST ONLY be sent if a control channel exists between a PU and PE. After communication has been established between a PE and PU, a new ASAP_BUSINESS_CARD message may be sent at any time by either entity to update its failover order.

The ASAP_BUSINESS_CARD message serves two purposes. First, it lists the pool handle. For a PU that is part of a PU/PE combination that is contacting a PE, this is essential so that the PE learns the pool handle of the PU/PE combination requesting service. Secondly, the ASAP_BUSINESS_CARD message tells the receiving entity a failover order that is recommended to follow. This should facilitate rendezvous between entities that have been working together, as well as to control the load redistribution upon the failure of any PE.

Upon receipt of an ASAP_BUSINESS_CARD message (see Section 2.2.13), the receiving ASAP Endpoint SHOULD:

BC1) Unpack the message, and if no entry exists in the translation cache of the receiving ASAP Endpoint for the pool handle listed within the ASAP_BUSINESS_CARD message, perform an ASAP_HANDLE_RESOLUTION for that pool handle. If the translation cache does hold an entry for the pool handle, then it may be necessary to update the peer endpoint.

BC2) Unpack the message and populate a preferred list for failover order. If the peer's PE should fail, this preferred list will be used to guide the ASAP Endpoint in the selection of an alternate PE.

4. Roles of Endpoints

A PU MUST implement the handling of ASAP_HANDLE_RESOLUTION and ASAP_HANDLE_RESOLUTION_RESPONSE messages. Furthermore, it MUST support the handling of ASAP_ERROR messages. It MAY implement the handling of ASAP_COOKIE, ASAP_COOKIE_ECHO, and ASAP_BUSINESS_CARD messages. It MAY also implement the handling of ASAP_SERVER_ANNOUNCE messages.

A PE MUST implement the handling of ASAP_REGISTRATION, ASAP_DEREGISTRATION, ASAP_REGISTRATION_RESPONSE, and ASAP_DEREGISTRATION_RESPONSE messages. Furthermore, it MUST support the handling of ASAP_ENDPOINT_KEEP_ALIVE, ASAP_ENDPOINT_KEEP_ALIVE_ACK, ASAP_ENDPOINT_UNREACHABLE, and ASAP_ERROR messages. It SHOULD support the handling of ASAP_COOKIE, ASAP_COOKIE_ECHO, and ASAP_BUSINESS_CARD messages. Furthermore, it MAY support the handling of ASAP_SERVER_ANNOUNCE messages.

An ENRP server MUST implement the handling of ASAP_REGISTRATION, ASAP_DEREGISTRATION, ASAP_REGISTRATION_RESPONSE, and ASAP_DEREGISTRATION_RESPONSE messages. Furthermore, it MUST support the handling of ASAP_ENDPOINT_KEEP_ALIVE, ASAP_ENDPOINT_KEEP_ALIVE_ACK, ASAP_ENDPOINT_UNREACHABLE, and ASAP_ERROR messages. Furthermore, it MAY support the handling of ASAP_SERVER_ANNOUNCE messages.

If a node acts as a PU and a PE, it MUST fulfill both roles.

5. SCTP Considerations

Each ASAP message is considered as an SCTP user message. The PPID registered for ASAP SHOULD be used. The SCTP port used at the ENRP server might be preconfigured or announced in the ASAP_SERVER_ANNOUNCE message or the well-known ASAP port.

ASAP messages belonging to the control channel MUST be sent using the PPID registered for ASAP. Messages belonging to the data channel MUST NOT use the PPID registered for ASAP.

6. The ASAP Interfaces

This chapter will focus primarily on the primitives and notifications that form the interface between the ASAP User and ASAP and that between ASAP and its lower-layer transport protocol (e.g., SCTP).

Note, the following primitive and notification descriptions are shown for illustrative purposes. We believe that including these descriptions in this document is important to the understanding of the operation of many aspects of ASAP; but an ASAP implementation is not required to use the exact syntax described in this section.

An ASAP User passes primitives to the ASAP sub-layer to request certain actions. Upon the completion of those actions or upon the detection of certain events, the ASAP layer will notify the ASAP User.

6.1. Registration.Request Primitive

```
      Format: registration.request(Pool Handle,
                                   User Transport parameter(s))
```

The Pool Handle parameter contains a NULL terminated ASCII string of fixed length. The optional User Transport parameter(s) indicates specific transport parameters and types with which to register. If this optional parameter is left off, then the SCTP endpoint used to communicate with the ENRP server is used as the default User Transport parameter. Note that any IP address contained within a User Transport parameter MUST be a bound IP address in the SCTP endpoint used to communicate with the ENRP server.

The ASAP User invokes this primitive to add itself to the handlespace, thus becoming a Pool Element of a pool. The ASAP User must register itself with the ENRP server by using this primitive before other ASAP Users using the handlespace can send message(s) to this ASAP User by Pool Handle or by PE handle (see Sections 6.5.1 and 6.5.3).

In response to the registration primitive, the ASAP Endpoint will send an ASAP_REGISTRATION message to the Home ENRP server (see Sections 2.2.1 and 3.1), and start a T2-registration timer.

6.2. Deregistration.Request Primitive

```
      Format: deregistration.request(Pool Handle)
```

The ASAP PE invokes this primitive to remove itself from the Server Pool. This should be used as a part of the graceful shutdown process by the application.

An ASAP_DEREGISTRATION message will be sent by the ASAP Endpoint to the Home ENRP server (see Sections 2.2.2 and 3.2).

6.3. CachePopulateRequest Primitive

```
Format: cache_populate_request([Pool-Handle |
                                Pool-Element-Handle])
```

If the address type is a Pool Handle and a local handle translation cache exists, the ASAP Endpoint should initiate a mapping information query by sending an ASAP_HANDLE_RESOLUTION message on the Pool handle and updating its local cache when the response comes back from the ENRP server.

If a Pool-Element-Handle is passed, then the Pool Handle is unpacked from the Pool-Element-Handle and the ASAP_HANDLE_RESOLUTION message is sent to the ENRP server for resolution. When the response message returns from the ENRP server, the local cache is updated.

Note that if the ASAP service does NOT support a local cache, this primitive performs NO action.

6.4. CachePurgeRequest Primitive

```
Format: cache_purge_request([Pool-Handle | Pool-Element-Handle])
```

If the user passes a Pool Handle and local handle translation cache exists, the ASAP Endpoint should remove the mapping information on the Pool Handle from its local cache. If the user passes a Pool-Element-Handle, then the Pool Handle within is used for the cache_purge_request.

Note that if the ASAP service does NOT support a local cache, this primitive performs NO action.

6.5. DataSendRequest Primitive

```
Format: data_send_request(destinationAddress, typeOfAddress,
                          message, sizeOfMessage, Options);
```

This primitive requests ASAP to send a message to some specified Pool or Pool Element within the current Operational scope.

Depending on the address type used for the send request, the sender’s ASAP Endpoint may perform address translation and Pool Element selection before sending the message out. This MAY also dictate the creation of a local transport endpoint in order to meet the required transport type.

The data_send_request primitive can take different forms of address types, as described in the following sections.

6.5.1. Sending to a Pool Handle

In this case, the destinationAddress and typeOfAddress together indicate a pool handle.

This is the simplest form of send_data_request primitive. By default, this directs ASAP to send the message to one of the Pool Elements in the specified pool.

Before sending the message out to the pool, the sender's ASAP endpoint MUST first perform a pool handle to address translation. It may also need to perform Pool Element selection if multiple Pool Elements exist in the pool.

If the sender's ASAP implementation does not support a local cache of the mapping information, or if it does not have the mapping information on the pool in its local cache, it will transmit an ASAP_HANDLE_RESOLUTION message (see Sections 2.2.5 and 3.3) to the current Home ENRP server and MUST hold the outbound message in queue while awaiting the response from the ENRP server (any further send request to this pool before the ENRP server responds SHOULD also be queued).

Once the necessary mapping information arrives from the ENRP server, the sender's ASAP will:

A) map the pool handle into a list of transport addresses of the destination PE(s);

B) if multiple PEs exist in the pool, choose one of them and transmit the message to it. In that case, the choice of the PE is made by the ASAP Endpoint of the sender based on the server pooling policy, as discussed in Section 6.5.2;

C) optionally create any transport endpoint that may be needed to communicate with the PE selected;

D) if no transport association or connection exists towards the destination PE, establish any needed transport state;

E) send out the queued message(s) to the appropriate transport connection using the appropriate send mechanism (e.g., for SCTP, the SEND primitive in [RFC4960] would be used); and,

F) if the local cache is implemented, append/update the local cache with the mapping information received in the ENRP server's response. Also, record the local transport information (e.g., the SCTP association id) if any new transport state was created.

For more on the ENRP server request procedures see [RFC5353].

Optionally, the ASAP Endpoint of the sender may return a Pool Element handle of the selected PE to the application after sending the message. This PE handle can then be used for future transmissions to that same PE (see Section 6.5.3).

Section 3.7 defines the failover procedures for cases where the selected PE is found unreachable.

6.5.2. Pool Element Selection

Each time an ASAP User sends a message to a pool that contains more than one PE, the sender’s ASAP Endpoint must select one of the PEs in the pool as the receiver of the current message. The selection is made according to the current server pooling policy of the pool to which the message is sent.

Note, no selection is needed if the ASAP_SEND_TOALL option is set (see Section 6.5.5).

Together with the server pooling policy, each PE can also specify a Policy Value for itself at the registration time. The meaning of the Policy Value depends on the current server pooling policy of the group. A PE can also change its Policy Value whenever it desires, by re-registering itself with the handlespace with a new Policy Value. Re-registration shall be done by simply sending another ASAP_REGISTRATION to its Home ENRP server (see Section 2.2.1).

One basic policy is defined in this document; others can be found in [RFC5356]

6.5.2.1. Round-Robin Policy

When an ASAP Endpoint sends messages by Pool Handle and Round-Robin is the current policy of that Pool, the ASAP Endpoint of the sender will select the receiver for each outbound message by Round-Robining through all the registered PEs in that Pool, in an attempt to achieve an even distribution of outbound messages. Note that in a large server pool, the ENRP server might not send back all PEs to the ASAP client. In this case, the client or PU will be performing a Round-Robin policy on a subset of the entire Pool.

6.5.3. Sending to a Pool Element Handle

In this case, the destinationAddress and typeOfAddress together indicate an ASAP Pool Element handle.

This requests that the ASAP Endpoint deliver the message to the PE identified by the Pool Element handle.

The Pool Element handle should contain the Pool Handle and a destination transport address of the destination PE or the Pool Handle and the transport type. Other implementation dependent elements may also be cached in a Pool Element handle.

The ASAP Endpoint shall use the transport address and transport type to identify the endpoint with which to communicate. If no communication state exists with the peer endpoint (and is required by the transport protocol), the ASAP Endpoint MAY set up the needed state and then invoke the SEND primitive for the particular transport protocol to send the message to the PE.

In addition, if a local translation cache is supported, the endpoint will:

A) send out the message to the transport address (or association id) designated by the PE handle.

B) determine if the Pool Handle is in the local cache.

   If it is *not*, the endpoint will:

   i) ask the Home ENRP server for handle resolution on the pool handle by sending an ASAP_HANDLE_RESOLUTION message (see Section 2.2.5), and

   ii) use the response to update the local cache.

      If the pool handle is in the cache, the endpoint will only update the pool handle if the cache is stale. A stale cache is indicated by it being older than the protocol parameter 'stale.cache.value' (see Section 7.2).

Sections 3.5 and 6.9 define the failover procedures for cases where the PE pointed to by the Pool Element handle is found to be unreachable.

Optionally, the ASAP Endpoint may return the actual Pool Element handle to which the message was sent (this may be different from the Pool Element handle specified when the primitive is invoked, due to the possibility of automatic failover).

6.5.4. Send by Transport Address

In this case, the destinationAddress and typeOfAddress together indicate a transport address and transport type.

This directs the sender's ASAP Endpoint to send the message out to the specified transport address.

No endpoint failover is supported when this form of send request is used. This form of send request effectively bypasses the ASAP endpoint.

6.5.5. Message Delivery Options

The Options parameter passed in the various forms of the above data_send_request primitive gives directions to the sender's ASAP endpoint on special handling of the message delivery.

The value of the Options parameter is generated by bit-wise "OR"ing of the following pre-defined constants:

ASAP_USE_DEFAULT: 0x0000 Use default setting.

ASAP_SEND_FAILOVER: 0x0001 Enables PE failover on this message. In the case where the first selected PE or the PE pointed to by the PE handle is found unreachable, the sender's ASAP Endpoint SHOULD re-select an alternate PE from the same pool if one exists, and silently re-send the message to this newly selected endpoint.

Note that this is a best-effort service. Applications should be aware that messages can be lost during the failover process, even if the underlying transport supports retrieval of unacknowledged data (e.g., SCTP). (Example: messages acknowledged by the SCTP layer at a PE, but not yet read by the PE when a PE failure occurs.) In the case where the underlying transport does not support such retrieval (e.g., TCP), any data already submitted by ASAP to the transport layer may be lost upon failover.

ASAP_SEND_NO_FAILOVER: 0x0002 This option prohibits the sender's ASAP Endpoint from re-sending the message to any alternate PE in case that the first selected PE, or the PE pointed to by the PE handle, is found to be unreachable. Instead, the sender's ASAP Endpoint shall notify its upper layer about the unreachability with an Error.Report and return any unsent data.

ASAP_SEND_TO_LAST: 0x0004 This option requests that the sender's ASAP Endpoint send the message to the same PE in the pool to which the previous message destined to this pool was sent.

ASAP_SEND_TO_ALL: 0x0008 When sending by Pool Handle, this option directs the sender's ASAP endpoint to send a copy of the message to all the PEs, except for the sender itself if the sender is a PE in that pool.

ASAP_SEND_TO_SELF: 0x0010 This option only applies in combination with the ASAP_SEND_TO_ALL option. It permits the sender's ASAP Endpoint to also deliver a copy of the message to itself if the sender is a PE of the pool (i.e., loop-back).

ASAP_SCTP_UNORDER: 0x1000 This option requests that the transport layer send the current message using un-ordered delivery (note the underlying transport must support un-ordered delivery for this option to be effective).

6.6. Data.Received Notification

```
Format: data.received(messageReceived, sizeOfMessage,
                      senderAddress, typeOfAddress)
```

When a new user message is received, the ASAP Endpoint of the receiver uses this notification to pass the message to its upper layer.

Along with the message being passed, the ASAP Endpoint of the receiver should also indicate to its upper layer the message senders address. The sender's address can be in the form of either an SCTP association id, TCP transport address, UDP transport address, or an ASAP Pool Element handle.

A) If the handle translation local cache is implemented at the receiver's ASAP Endpoint, a reverse mapping from the sender's IP address to the pool handle should be performed, and if the mapping is successful, the sender's ASAP Pool Element handle should be constructed and passed in the senderAddress field.

B) If there is no local cache or the reverse mapping is not successful, the SCTP association id or other transport specific identification (if SCTP is not being used) should be passed in the senderAddress field.

6.7. Error.Report Notification

```
Format: error.report(destinationAddress, typeOfAddress,
                     failedMessage, sizeOfMessage)
```

An error.report should be generated to notify the ASAP User about failed message delivery as well as other abnormalities.

The destinationAddress and typeOfAddress together indicate to whom the message was originally sent. The address type can be either an ASAP Pool Element handle, association id, or a transport address.

The original message (or the first portion of it if the message is too big) and its size should be passed in the failedMessage and sizeOfMessage fields, respectively.

6.8. Examples

These examples assume an underlying SCTP transport between the PE and PU. Other transports are possible, but SCTP is utilized in the examples for illustrative purposes. Note that all communication between the PU and ENRP server and the PE and ENRP servers would be using SCTP.

6.8.1. Send to a New Pool

This example shows the event sequence when a Pool User sends the message "hello" to a pool that is not in the local translation cache (assuming local caching is supported).

```
ENRP Server                       PU         new-handle:PEx

  |                                |                 |
  |                              +---+               |
  |                              | 1 |               |
  |2. ASAP_HANDLE_RESOLUTION     +---+               |
  |<-------------------------------|                 |
  |                              +---+               |
  |                              | 3 |               |
  |4. ASAP_HANDLE_RESOLUTION_RSP +---+               |
  |------------------------------->|                 |
  |                              +---+               |
  |                              | 5 |               |
  |                              +---+  6. "hello1"  |
  |                                |---------------->|
  |                                |                 |
```

1) The user at PU invokes:

   data_send_request("new-handle", handle-type, "hello1", 6, 0);

   The ASAP Endpoint, in response, looks up the pool "new-handle" in its local cache, but fails to find it.

2) The ASAP Endpoint of the PU queues the message and sends an ASAP_HANDLE_RESOLUTION request to the ENRP server asking for all information about pool "new-handle".

3) A T1-ENRPrequest timer is started while the ASAP Endpoint is waiting for the response from the ENRP server.

4) The ENRP server responds to the query with an ASAP_HANDLE_RESOLUTION_RESPONSE message that contains all the information about pool "new-handle".

5) ASAP at PU cancels the T1-ENRPrequest timer and populate its local cache with information on pool "new-handle".

6) Based on the server pooling policy of pool "new-handle", ASAP at PU selects the destination PE (PEx), sets up, if necessary, an SCTP association towards PEx (explicitly or implicitly), and sends out the queued "hello1" user message.

6.8.2. Send to a Cached Pool Handle

This shows the event sequence when the ASAP User PU sends another message to the pool "new-handle" after what happened in Section 6.8.1.

```
  ENRP Server                        PU         new-handle:PEx

     |                                |                  |
     |                              +---+                |
     |                              | 1 |                |
     |                              +---+  2. "hello2"   |
     |                                |----------------->|
     |                                |                  |
```

1) The user at PU invokes:

   data_send_request("new-handle", handle-type, "hello2", 6, 0);

   The ASAP Endpoint, in response, looks up the pool "new-handle" in its local cache and finds the mapping information.

2) Based on the server pooling policy of "new-handle", ASAP at PU selects the PE (assuming EPx is selected again), and sends out "hello2" message (assuming the SCTP association is already set up).

6.9. PE Send Failure

When the ASAP Endpoint in a PE or PU attempts to send a message to a PE and fails, the failed sender will report the event as described in Section 3.5.

Additional primitives are also defined in this section to support those user applications that do not wish to use ASAP as the actual transport.

6.9.1. Translation.Request Primitive

Format: translation.request(Pool-Handle)

If the address type is a Pool Handle and a local handle translation cache exists, the ASAP Endpoint should look within its translation cache and return the current known transport types, ports, and addresses to the caller.

If the Pool Handle does not exist in the local handle cache or no handle cache exists, the ASAP Endpoint will send an ASAP_HANDLE_RESOLUTION request using the Pool Handle. Upon completion of the handle resolution, the ASAP Endpoint should populate the local handle cache (if a local handle cache is supported) and return the transport types, ports, and addresses to the caller.

6.9.2. Transport.Failure Primitive

Format: transport.failure(Pool-Handle, Transport-address)

If an external user encounters a failure in sending to a PE and is *not* using ASAP, it can use this primitive to report the failure to the ASAP endpoint. ASAP will send an ASAP_ENDPOINT_UNREACHABLE to the "Home" ENRP server in response to this primitive. Note ASAP SHOULD NOT send an ASAP_ENDPOINT_UNREACHABLE *unless* the user has actually made a previous request to send data to the PE.

7. Timers, Variables, and Thresholds

The following is a summary of the timers, variables, and pre-set protocol constants used in ASAP.

7.1. Timers

T1-ENRPrequest - A timer started when a request is sent by ASAP to the ENRP server (providing application information is queued). Normally set to 15 seconds.

T2-registration - A timer started when sending an ASAP_REGISTRATION request to the Home ENRP server, normally set to 30 seconds.

T3-deregistration - A timer started when sending a de-registration request to the Home ENRP server, normally set to 30 seconds.

T4-reregistration - This timer is started after successful registration into the ENRP handlespace and is used to cause a re-registration at a periodic interval. This timer is normally set to 10 minutes or 20 seconds less than the Lifetime parameter used in the registration request (whichever is less).

T5-Serverhunt - This timer is used during the ENRP Server Hunt procedure and is normally set to 10 seconds.

T6-Serverannounce - This timer gives the time between the sending of consecutive ASAP_SERVER_ANNOUNCE messages. It is normally set to 1 second.

T7-ENRPoutdate - This timer gives the time a server announcement is valid. It is normally set to 5 seconds.

7.2. Variables

stale_cache_value - A threshold variable that indicates how long a cache entry is valid for.

7.3. Thresholds

MAX-REG-ATTEMPT - The maximum number of registration attempts to be made before a server hunt is issued. The default value of this is set to 2.

MAX-REQUEST-RETRANSMIT - The maximum number of attempts to be made when requesting information from the local ENRP server before a server hunt is issued. The default value for this is 2.

RETRAN-MAX - This value represents the maximum time between registration attempts and puts a ceiling on how far the registration timer will back off. The default value for this is normally set to 60 seconds.

8. IANA Considerations

This document (RFC 5352) is the reference for all registrations described in this section. All registrations have been listed on the Reliable Server Pooling (RSerPool) Parameters page.

8.1. A New Table for ASAP Message Types

ASAP Message Types are maintained by IANA. Fourteen initial values have been assigned by IANA as described in Figure 1. IANA created a new table, "ASAP Message Types":

Type	Message Name	Reference
0x00	(Reserved by IETF)	RFC 5352
0x01	ASAP_REGISTRATION	RFC 5352
0x02	ASAP_DEREGISTRATION	RFC 5352
0x03	ASAP_REGISTRATION_RESPONSE	RFC 5352
0x04	ASAP_DEREGISTRATION_RESPONSE	RFC 5352
0x05	ASAP_HANDLE_RESOLUTION	RFC 5352
0x06	ASAP_HANDLE_RESOLUTION_RESPONSE	RFC 5352
0x07	ASAP_ENDPOINT_KEEP_ALIVE	RFC 5352
0x08	ASAP_ENDPOINT_KEEP_ALIVE_ACK	RFC 5352
0x09	ASAP_ENDPOINT_UNREACHABLE	RFC 5352
0x0a	ASAP_SERVER_ANNOUNCE	RFC 5352
0x0b	ASAP_COOKIE	RFC 5352
0x0c	ASAP_COOKIE_ECHO	RFC 5352
0x0d	ASAP_BUSINESS_CARD	RFC 5352
0x0e	ASAP_ERROR	RFC 5352
0x0b-0xff	(Available for Assignment)	RFC 5352

Requests to register an ASAP Message Type in this table should be sent to IANA. The number must be unique. The "Specification Required" policy of [RFC5226] MUST be applied.

8.2. Port Numbers

The references for the already assigned port numbers

asap-tcp 3863/tcp

asap-udp 3863/udp

asap-sctp 3863/sctp

asap-tcp-tls 3864/tcp

asap-sctp-tls 3864/sctp

have been updated to RFC 5352.

8.3. SCTP Payload Protocol Identifier

The reference for the already assigned ASAP payload protocol identifier 11 has been updated to RFC 5352.

8.4. Multicast Addresses

IANA has assigned an IPv4 multicast address (224.0.1.185) and an IPv6 multicast address (FF0X:0:0:0:0:0:0:133). The IPv4 address is part of the Internetwork Control Block (224.0.1/24).

9. Security Considerations

We present a summary of the of the threats to the RSerPool architecture and describe security requirements in response in order to mitigate the threats. Next, we present the security mechanisms, based on TLS, that are implementation requirements in response to the threats. Finally, we present a chain-of-trust argument that examines critical data paths in RSerPool and shows how these paths are protected by the TLS implementation.

9.1. Summary of RSerPool Security Threats

"Threats Introduced by Reliable Server Pooling (RSerPool) and Requirements for Security in Response to Threats" [RFC5355] describes the threats to the RSerPool architecture in detail and lists the security requirements in response to each threat. From the threats described in this document, the security services required for the RSerPool protocol are enumerated below.

Threat 1) PE registration/de-registration flooding or spoofing.

Security mechanism in response: ENRP server authenticates the PE.

Threat 2) PE registers with a malicious ENRP server.

Security mechanism in response: PE authenticates the ENRP server.

Threats 1 and 2, taken together, result in mutual authentication of the ENRP server and the PE.

Threat 3) Malicious ENRP server joins the ENRP server pool.

Security mechanism in response: ENRP servers mutually authenticate.

Threat 4) A PU communicates with a malicious ENRP server for handle resolution.

Security mechanism in response: The PU authenticates the ENRP server.

Threat 5) Replay attack.

Security mechanism in response: Security protocol that has protection from replay attacks.

Threat 6) Corrupted data that causes a PU to have misinformation concerning a pool handle resolution.

Security mechanism in response: Security protocol that supports integrity protection.

Threat 7) Eavesdropper snooping on handlespace information.

Security mechanism in response: Security protocol that supports data confidentiality.

Threat 8) Flood of ASAP_ENDPOINT_UNREACHABLE messages from the PU to ENRP server.

Security mechanism in response: ASAP must control the number of ASAP Endpoint unreachable messages transmitted from the PU to the ENRP server.

Threat 9) Flood of ASAP_ENDPOINT_KEEP_ALIVE messages to the PE from the ENRP server.

Security mechanism in response: ENRP server must control the number of ASAP_ENDPOINT_KEEP_ALIVE messages to the PE.

To summarize, the threats 1-7 require security mechanisms that support authentication, integrity, data confidentiality, and protection from replay attacks.

For RSerPool we need to authenticate the following:

```
PU <----  ENRP server (PU authenticates the ENRP server)
PE <----> ENRP server (mutual authentication)
ENRP server <-----> ENRP server (mutual authentication)
```

9.2. Implementing Security Mechanisms

We do not define any new security mechanisms specifically for responding to threats 1-7. Rather, we use an existing IETF security protocol, specifically [RFC3237], to provide the security services required. TLS supports all these requirements and MUST be implemented. The TLS_RSA_WITH_AES_128_CBC_SHA ciphersuite MUST be supported, at a minimum, by implementers of TLS for RSerPool. For purposes of backwards compatibility, ENRP SHOULD support TLS_RSA_WITH_3DES_EDE_CBC_SHA. Implementers MAY also support any other IETF-approved ciphersuites.

ENRP servers, PEs, and PUs MUST implement TLS. ENRP servers and PEs MUST support mutual authentication using PSK (pre-shared-key). ENRP servers MUST support mutual authentication among themselves using PSK. PUs MUST authenticate ENRP servers using certificates.

TLS with PSK is mandatory to implement as the authentication mechanism for ENRP to ENRP authentication and PE to ENRP authentication. For PSK, having a pre-shared-key constitutes authorization. The network administrators of a pool need to decide which nodes are authorized to participate in the pool. The justification for PSK is that we assume that one administrative domain will control and manage the server pool. This allows for PSK to be implemented and managed by a central security administrator.

TLS with certificates is mandatory to implement as the authentication mechanism for PUs to the ENRP server. PUs MUST authenticate ENRP servers using certificates. ENRP servers MUST possess a site certificate whose subject corresponds to their canonical hostname. PUs MAY have certificates of their own for mutual authentication with TLS, but no provisions are set forth in this document for their use. All RSerPool Elements that support TLS MUST have a mechanism for validating certificates received during TLS negotiation; this entails possession of one or more root certificates issued by certificate authorities (preferably, well-known distributors of site certificates comparable to those that issue root certificates for web browsers).

In order to prevent man-in-the-middle attacks, the client MUST verify the server’s identity (as presented in the server’s Certificate message). The client’s understanding of the server’s identity (typically, the identity used to establish the transport connection) is called the "reference identity". The client determines the type (e.g., DNS name or IP address) of the reference identity and performs a comparison between the reference identity and each subjectAltName value of the corresponding type until a match is produced. Once a match is produced, the server’s identity has been verified, and the server identity check is complete. Different subjectAltName types are matched in different ways. The client may map the reference identity to a different type prior to performing a comparison. Mappings may be performed for all available subjectAltName types to which the reference identity can be mapped; however, the reference identity should only be mapped to types for which the mapping is either inherently secure (e.g., extracting the DNS name from a URI to compare with a subjectAltName of type dNSName) or for which the mapping is performed in a secure manner (e.g., using DNS Security (DNSSEC), or using user- or admin-configured host-to-address/ address-to-host lookup tables).

If the server identity check fails, user-oriented clients SHOULD either notify the user or close the transport connection and indicate that the server’s identity is suspect. Automated clients SHOULD close the transport connection and then return or log an error indicating that the server’s identity is suspect, or both. Beyond the server identity check described in this section, clients should be prepared to do further checking to ensure that the server is authorized to provide the service it is requested to provide. The client may need to make use of local policy information in making this determination.

If the reference identity is an internationalized domain name, conforming implementations MUST convert it to the ASCII Compatible Encoding (ACE) format, as specified in Section 4 of [RFC3490], before comparison with subjectAltName values of type dNSName. Specifically, conforming implementations MUST perform the conversion operation specified in Section 4 of [RFC3490] as follows: * in step 1, the domain name SHALL be considered a "stored string"; * in step 3, set the flag called "UseSTD3ASCIIRules"; * in step 4, process each label with the "ToASCII" operation; and * in step 5, change all label separators to U+002E (full stop).

After performing the "to-ASCII" conversion, the DNS labels and names MUST be compared for equality, according to the rules specified in Section 3 of RFC 3490. The '*' (ASCII 42) wildcard character is allowed in subjectAltName values of type dNSName, and then, only as the left-most (least significant) DNS label in that value. This wildcard matches any left-most DNS label in the server name. That is, the subject *.example.com matches the server names a.example.com and b.example.com, but does not match example.com or a.b.example.com.

When the reference identity is an IP address, the identity MUST be converted to the "network byte order" octet string representation in [RFC0791] and [RFC2460]. For IP version 4, as specified in RFC 791, the octet string will contain exactly four octets. For IP version 6, as specified in RFC 2460, the octet string will contain exactly sixteen octets. This octet string is then compared against subjectAltName values of type iPAddress. A match occurs if the reference identity octet string and value octet strings are identical.

After a TLS layer is established in a session, both parties are to independently decide whether or not to continue based on local policy and the security level achieved. If either party decides that the security level is inadequate for it to continue, it SHOULD remove the TLS layer immediately after the TLS (re)negotiation has completed (see RFC 4511)[RFC4511]. Implementations may re-evaluate the security level at any time and, upon finding it inadequate, should remove the TLS layer.

Implementations MUST support TLS with SCTP, as described in [RFC3436] or TLS over TCP, as described in [RFC5246]. When using TLS/SCTP we must ensure that RSerPool does not use any features of SCTP that are not available to a TLS/SCTP user. This is not a difficult technical problem, but simply a requirement. When describing an API of the RSerPool lower layer, we also have to take into account the differences between TLS and SCTP.

Threat 8 requires the ASAP protocol to limit the number of ASAP_ENDPOINT_UNREACHABLE messages (see Section 3.5) to the ENRP server.

Threat 9 requires the ENRP protocol to limit the number of ASAP_ENDPOINT_KEEP_ALIVE messages from the ENRP server to the PE (see [RFC5353]).

There is no security mechanism defined for the multicast announcements. Therefore, a receiver of such an announcement cannot consider the source address of such a message to be a trustworthy address of an ENRP server. A receiver must also be prepared to receive a large number of multicast announcements from attackers.

9.3. Chain of Trust

Security is mandatory to implement in RSerPool and is based on TLS implementation in all three architecture components that comprise RSerPool -- namely PU, PE, and ENRP server. We define an ENRP server that uses TLS for all communication and authenticates ENRP peers and PE registrants to be a secured ENRP server.

Here is a description of all possible data paths and a description of the security.

```
PU <---> secured ENRP server (authentication of ENRP server;
         queries over TLS)
PE <---> secured ENRP server (mutual authentication;
         registration/de-registration over TLS)
secured ENRP server <---> secured ENRP server (mutual authentication;
         database updates using TLS)
```

If all components of the system authenticate and communicate using TLS, the chain of trust is sound. The root of the trust chain is the ENRP server. If that is secured using TLS, then security will be enforced for all ENRP and PE components that try to connect to it.

Summary of interaction between secured and unsecured components: If the PE does not use TLS and tries to register with a secure ENRP server, it will receive an error message response indicated as an error due to security considerations and the registration will be rejected. If an ENRP server that does not use TLS tries to update the database of a secure ENRP server, then the update will be rejected. If a PU does not use TLS and communicates with a secure ENRP server, it will get a response with the understanding that the response is not secure, as the response can be tampered with in transit even if the ENRP database is secured.

The final case is the PU sending a secure request to ENRP. It might be that ENRP and PEs are not secured and this is an allowable configuration. The intent is to secure the communication over the Internet between the PU and the ENRP server.

Summary:

RSerPool architecture components can communicate with each other to establish a chain of trust. Secured PE and ENRP servers reject any communications with unsecured ENRP or PE servers.

If the above is enforced, then a chain of trust is established for the RSerPool user.

10. Acknowledgments

The authors wish to thank John Loughney, Lyndon Ong, Walter Johnson, Thomas Dreibholz, and many others for their invaluable comments and feedback.

11. References

11.1. Normative References

[RFC0791] Postel, J., "Internet Protocol", STD 5, RFC 791, September 1981.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC2460] Deering, S. and R. Hinden, "Internet Protocol, Version 6 (IPv6) Specification", RFC 2460, December 1998.

[RFC3237] Tuexen, M., Xie, Q., Stewart, R., Shore, M., Ong, L., Loughney, J., and M. Stillman, "Requirements for Reliable Server Pooling", RFC 3237, January 2002.

[RFC3436] Jungmaier, A., Rescorla, E., and M. Tuexen, "Transport Layer Security over Stream Control Transmission Protocol", RFC 3436, December 2002.

[RFC3490] Faltstrom, P., Hoffman, P., and A. Costello, "Internationalizing Domain Names in Applications (IDNA)", RFC 3490, March 2003.

[RFC5246] Dierks, T. and E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.2", RFC 5246, August 2008.

[RFC4511] Sermersheim, J., "Lightweight Directory Access Protocol (LDAP): The Protocol", RFC 4511, June 2006.

[RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, May 2008.

[RFC5356] Dreibholz, T. and M. Tuexen, "Reliable Server Pooling Policies", RFC 5356, September 2008.

[RFC5354] Stewart, R., Xie, Q., Stillman, M., and M. Tuexen, "Aggregate Server Access Protocol (ASAP) and Endpoint Handlespace Redundancy Protocol (ENRP) Parameters", RFC 5354, September 2008.

[RFC5353] Xie, Q., Stewart, R., Stillman, M., Tuexen, M., and A. Silverton, "Endpoint Handlespace Redundancy Protocol (ENRP)", RFC 5353, September 2008.

[RFC5355] Stillman, M., Ed., Gopal, R., Guttman, E., Holdrege, M., and S. Sengodan, "Threats Introduced by Reliable Server Pooling (RSerPool) and Requirements for Security in Response to Threats", RFC 5355, September 2008.

11.2. Informative References

[RFC4086] Eastlake, D., Schiller, J., and S. Crocker, "Randomness Requirements for Security", BCP 106, RFC 4086, June 2005.

Authors' Addresses

Randall R. Stewart
The Resource Group
1700 Pennsylvania Ave NW
Suite 560
Washington, D.C., 20006
USA

EMail: randall@lakerest.net

Qiaobing Xie
The Resource Group
1700 Pennsylvania Ave NW
Suite 560
Washington, D.C., 20006
USA

Phone: +1 224-465-5954
EMail: Qiaobing.Xie@gmail.com

Maureen Stillman
Nokia
1167 Peachtree Ct.
Naperville, IL 60540
USA

EMail: maureen.stillman@nokia.com

Michael Tuexen
Muenster Univ. of Applied Sciences
Stegerwaldstr. 39
48565 Steinfurt
Germany

EMail: tuexen@fh-muenster.de

Full Copyright Statement

Copyright (C) The IETF Trust (2008).

This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY, THE IETF TRUST AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Intellectual Property

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

The IETF invites any interested party to bring to its attention any copyrights, patents or patent applications, or other proprietary rights that may cover technology that may be required to implement this standard. Please address the information to the IETF at ietf-ipr@ietf.org.