Network Working Group
Request for Comments: 5353
Category: Experimental

Q. Xie
R. Stewart
The Resource Group
M. Stillman
Nokia
M. Tuexen
Muenster Univ. of Applied Sciences
A. Silverton
Sun Microsystems, Inc.
September 2008

Endpoint Handlespace Redundancy Protocol (ENRP)

Status of This Memo

Abstract

The Endpoint Handlespace Redundancy Protocol (ENRP) is designed to work in conjunction with the Aggregate Server Access Protocol (ASAP) to accomplish the functionality of the Reliable Server Pooling (RSerPool) requirements and architecture. Within the operational scope of RSerPool, ENRP defines the procedures and message formats of a distributed, fault-tolerant registry service for storing, bookkeeping, retrieving, and distributing pool operation and membership information.

Table of Contents

1. Introduction

ENRP is designed to work in conjunction with ASAP [RFC5352] to accomplish the functionality of RSerPool as defined by its requirements [RFC3237].

Within the operational scope of RSerPool, ENRP defines the procedures and message formats of a distributed, fault-tolerant registry service for storing, bookkeeping, retrieving, and distributing pool operation and membership information.

Whenever appropriate, in the rest of this document, we will refer to this RSerPool registry service as ENRP handlespace, or simply handlespace, because it manages all pool handles.

1.1. Definitions

This document uses the following terms:

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.

Handle space: A cohesive structure of pool handles and relations that may be queried by an internal or external agent.

ENRP client channel: The communication channel through which an ASAP User (either a Pool Element (PE) or Pool User (PU)) requests ENRP handlespace service. The client channel is usually defined by the transport address of the Home ENRP server and a well-known port number.

ENRP server channel: Defined by a list of IP addresses (one for each ENRP server in an operational scope) and a well-known port number. All ENRP servers in an operational scope can send "group-cast" messages to other servers through this channel. In a "group-cast", the sending server sends multiple copies of the message, one to each of its peer servers, over a set of point-to-point Stream Control Transmission Protocol (SCTP) associations between the sending server and the peers. The "group-cast" may be conveniently implemented with the use of the "SCTP_SENDALL" option on a one-to-many style SCTP socket.

Home ENRP server: The ENRP server to which a PE or PU currently belongs. A PE MUST only have one Home ENRP server at any given time, and both the PE and its Home ENRP server MUST keep track of this master/slave relationship between them. A PU SHOULD select one of the available ENRP servers as its Home ENRP server.

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].

2. ENRP Message Definitions

In this section, we define the format of all ENRP messages. These are messages sent and received amongst ENRP servers in an operational scope. Messages sent and received between a PE/PU and an ENRP server are part of ASAP and are defined in [RFC5352]. A common format, that is defined in [RFC5354], is used for all ENRP and ASAP messages.

Most ENRP messages contain a combination of fixed fields and TLV (Type-Length-Value) parameters. The TLV parameters are also defined in [RFC5354]. If a nested TLV parameter is not ended on a 32-bit word boundary, it will be padded with all '0' octets to the next 32-bit word boundary.

All messages, as well as their fields/parameters described below, MUST be transmitted in network byte order (aka Big Endian, meaning the most significant byte is transmitted first).

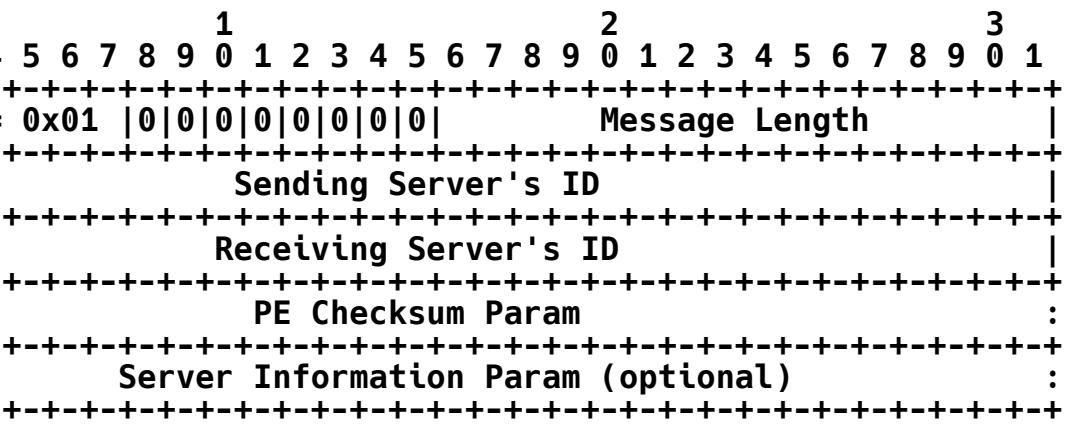
For ENRP, the following message types are defined in this section:

Type	Message Name
0x00	- (Reserved by IETF)
0x01	- ENRP_PRESENCE
0x02	- ENRP_HANDLE_TABLE_REQUEST
0x03	- ENRP_HANDLE_TABLE_RESPONSE
0x04	- ENRP_HANDLE_UPDATE
0x05	- ENRP_LIST_REQUEST
0x06	- ENRP_LIST_RESPONSE
0x07	- ENRP_INIT_TAKEOVER
0x08	- ENRP_INIT_TAKEOVER_ACK
0x09	- ENRP_TAKEOVER_SERVER
0x0a	- ENRP_ERROR
0x0b-0xff	- (Reserved by IETF)

Figure 1

2.1. ENRP_PRESENCE Message

This ENRP message is used to announce (periodically) the presence of an ENRP server, or to probe the status of a peer ENRP server. This message is either sent on the ENRP server channel or sent point-to-point to another ENRP server.



Sending Server's ID: 32 bits (unsigned integer)

This is the ID of the ENRP server that sent this message.

Receiving Server's ID: 32 bits (unsigned integer)

This is the ID of the ENRP server to which this message is intended. If the message is not intended for an individual server (e.g., the message is group-casted to a group of servers), this field MUST be sent with all 0s. If the message is sent point-to-point, this field MAY be sent with all 0s.

PE Checksum Parameter:

This is a TLV that contains the latest PE checksum of the ENRP server that sends the ENRP_PRESENCE. This parameter SHOULD be included for handlespace consistency auditing. See Section 3.6.1 for details.

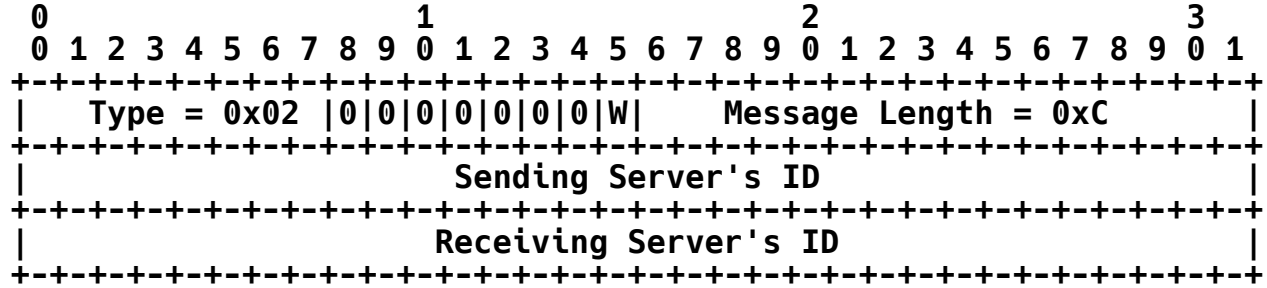
Server Information Parameter:

If this parameter is present, it contains the server information of the sender of this message (the Server Information Parameter is defined in [RFC5354]). This parameter is optional. However, if this message is sent in response to a received "reply required" ENRP_PRESENCE from a peer, the sender then MUST include its server information.

Note, at startup, an ENRP server MUST pick a randomly generated, non-zero 32-bit unsigned integer as its ID and MUST use this same ID until the ENRP server is rebooted.

2.2. ENRP_HANDLE_TABLE_REQUEST Message

An ENRP server sends this message to one of its peers to request a copy of the handlespace data. This message is normally used during server initialization or handlespace re-synchronization.



W (oWn-children-only) Flag: 1 bit

Set to '1' if the sender of this message is only requesting information about the PEs owned by the message receiver. Otherwise, set to '0'.

Sending Server's ID:

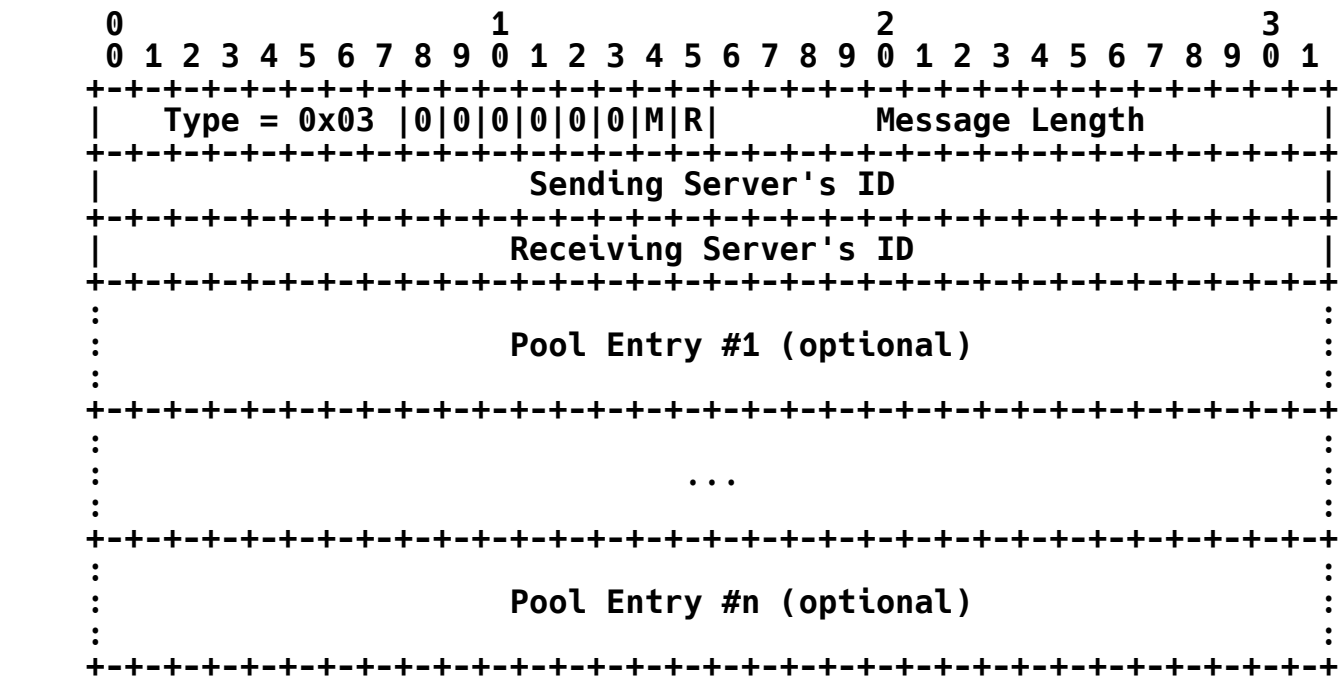
See Section 2.1.

Receiving Server's ID:

See Section 2.1.

2.3. ENRP_HANDLE_TABLE_RESPONSE Message

The PEER_NAME_TABLE_RESPONSE message is sent by an ENRP server in response to a received PEER_NAME_TABLE_REQUEST message to assist peer-server initialization or handlespace synchronization.



M (More_to_send) Flag: 1 bit

Set to '1' if the sender of this message has more pool entries to send in subsequent ENRP_HANDLE_TABLE_RESPONSE messages. Otherwise, set to '0'.

R (Reject) Flag: 1 bit

MUST be set to '1' if the sender of this message is rejecting a handlespace request. In this case, pool entries MUST NOT be included. This might happen if the sender of this message is in the middle of initializing its database or is under high load.

Message Length: 16 bits (unsigned integer)

Indicates the entire length of the message, including the header, in number of octets.

Note, the value in the Message Length field will NOT cover any padding at the end of this message.

Sending Server's ID:

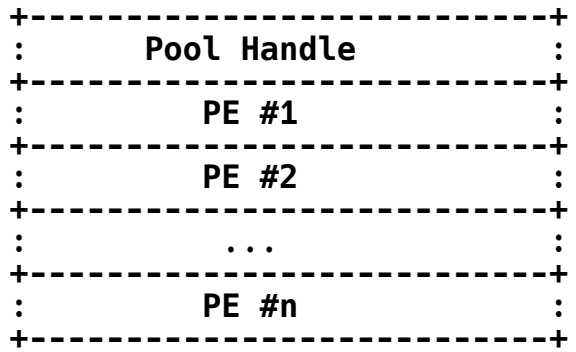
See Section 2.1.

Receiving Server's ID:

See Section 2.1.

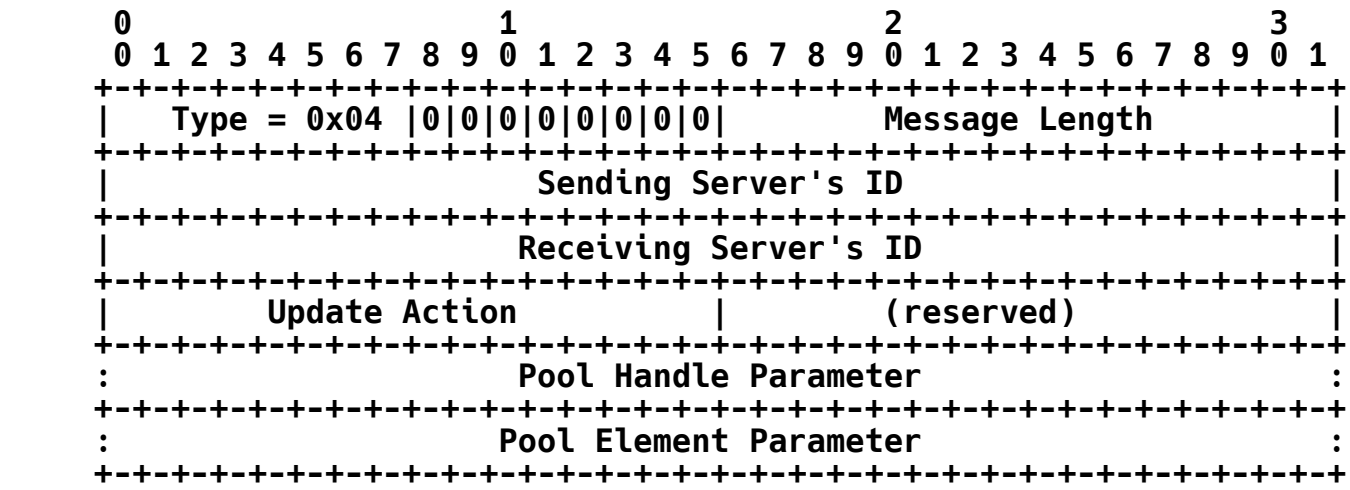
Pool Entry #1-#n:

If the R flag is set to '0', at least one pool entry SHOULD be present in this message. Each pool entry MUST start with a Pool Handle parameter, as defined in Section 3.9 of [RFC5354], and is followed by one or more Pool Element parameters in TLV format, as shown below:



2.4. ENRP_HANDLE_UPDATE Message

The PEER_NAME_UPDATE message is sent by the Home ENRP server of a PE to all peer servers to announce registration, re-registration, or de-registration of the PE in the handlespace.



Message Length: 16 bits (unsigned integer)

Indicates the entire length of the message, including the header, in number of octets.

Note, the value in the Message Length field will NOT cover any padding at the end of this message.

Update Action: 16 bits (unsigned integer)

This field indicates the requested action of the specified PE. The field MUST be set to one of the following values:

0x0000 - ADD_PE: Add or update the specified PE in the ENRP handlespace.

0x0001 - DEL_PE: Delete the specified PE from the ENRP handlespace.

0x0002 - 0xFFFF: Reserved by IETF.

Other values are reserved by IETF and MUST NOT be used.

Reserved: 16 bits

This field MUST be set to all 0s by the sender and ignored by the receiver.

Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

See Section 2.1.

Pool Handle:

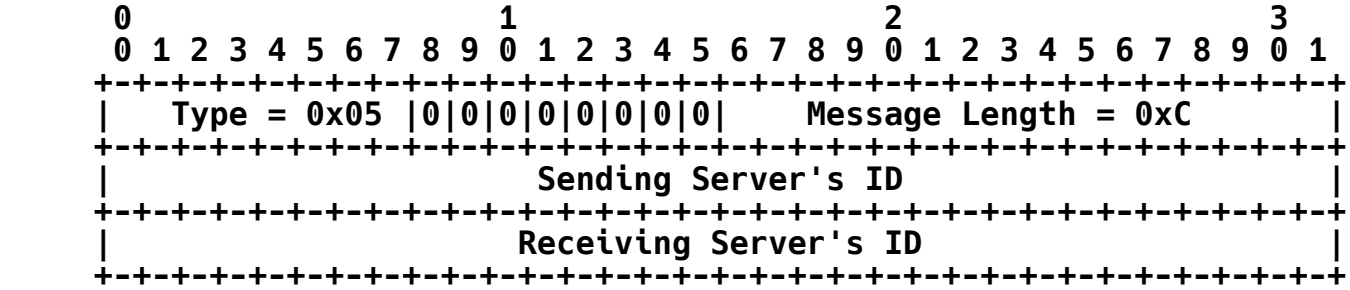
Specifies to which the PE belongs.

Pool Element:

Specifies the PE.

2.5. ENRP_LIST_REQUEST Message

The PEER_LIST_REQUEST message is sent to request a current copy of the ENRP server list. This message is normally sent from a newly activated ENRP server to an established ENRP server as part of the initialization process.



Sending Server's ID:

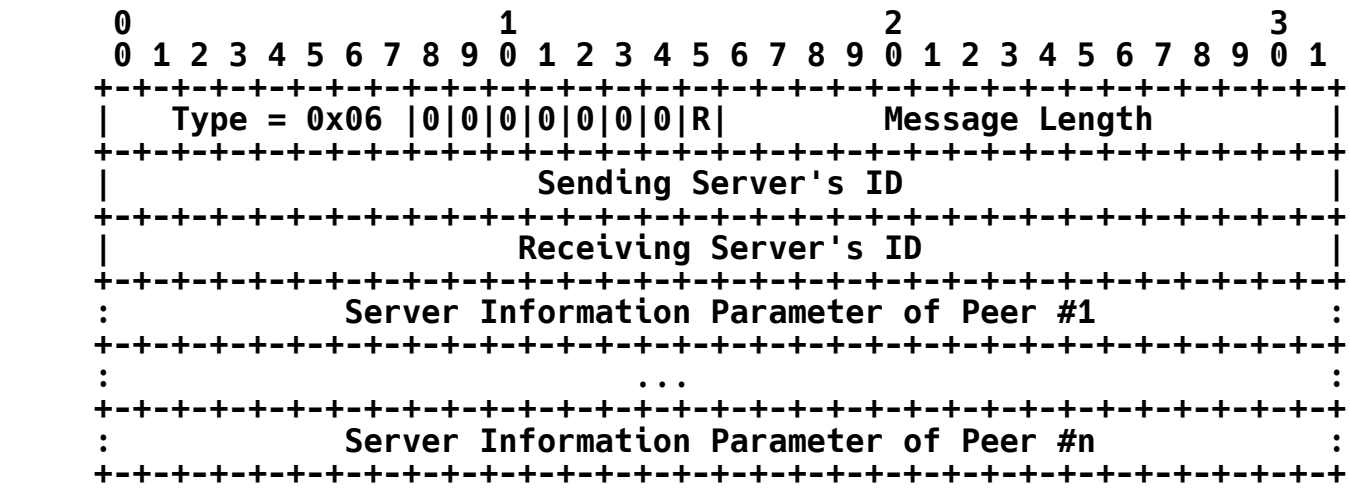
See Section 2.1.

Receiving Server's ID:

See Section 2.1.

2.6. ENRP_LIST_RESPONSE Message

The PEER_LIST_RESPONSE message is sent in response from an ENRP server that receives a PEER_LIST_REQUEST message to return information about known ENRP servers.



R (Reject) Flag: 1 bit

This flag MUST be set to '1' if the sender of this message is rejecting a PEER_LIST_REQUEST message. If this case occurs, the message MUST NOT include any Server Information Parameters.

Message Length: 16 bits (unsigned integer)

Indicates the entire length of the message in number of octets.

Note, the value in the Message Length field will NOT cover any padding at the end of this message.

Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

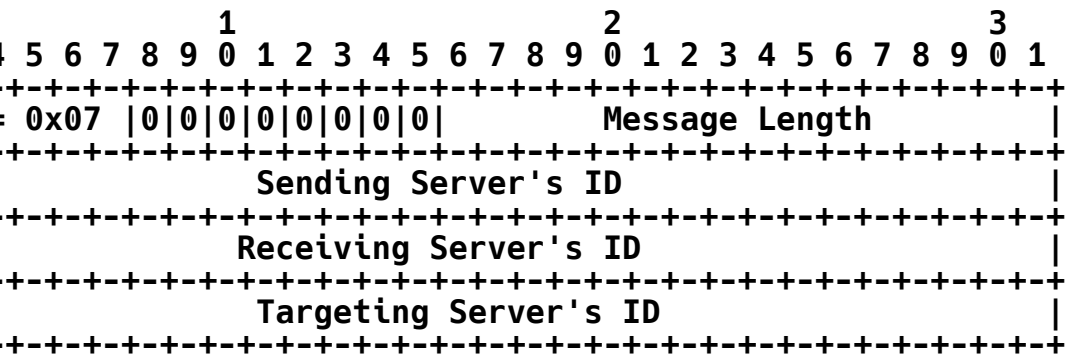
See Section 2.1.

Server Information Parameter of Peer #1-#n:

Each contains a Server Information Parameter of a peer known to the sender. The Server Information Parameter is defined in [RFC5354].

2.7. ENRP_INIT_TAKEOVER Message

The ENRP_INIT_TAKEOVER message is sent by an ENRP server (the takeover initiator) to announce its intention of taking over a specific peer ENRP server. It is sent to all its peers.



Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

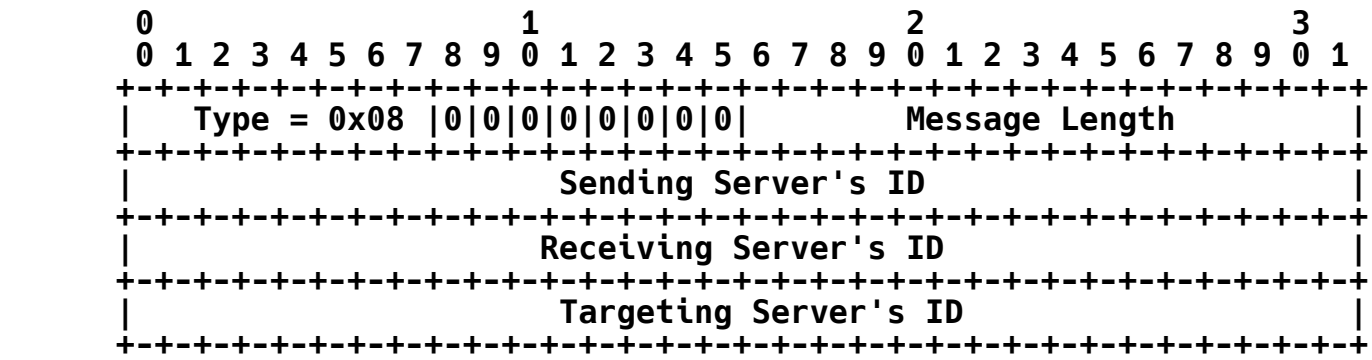
See Section 2.1.

Targeting Server's ID: 32 bits (unsigned integer)

This is the ID of the peer ENRP that is the target of this takeover attempt.

2.8. ENRP_INIT_TAKEOVER_ACK Message

The PEER_INIT_TAKEOVER_ACK message is sent in response to a takeover initiator to acknowledge the reception of the PEER_INIT_TAKEOVER message and that it does not object to the takeover.



Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

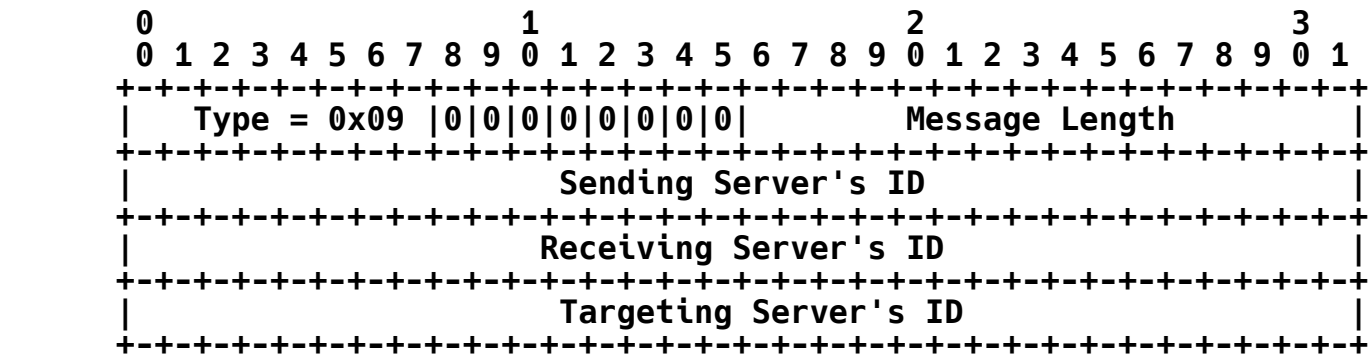
See Section 2.1.

Targeting Server's ID:

This is the ID of the peer ENRP that is the target of this takeover attempt.

2.9. ENRP_TAKEOVER_SERVER Message

The PEER_TAKEOVER_REGISTRAR message is sent by the takeover initiator to declare the enforcement of a takeover to all active peer ENRP servers.



Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

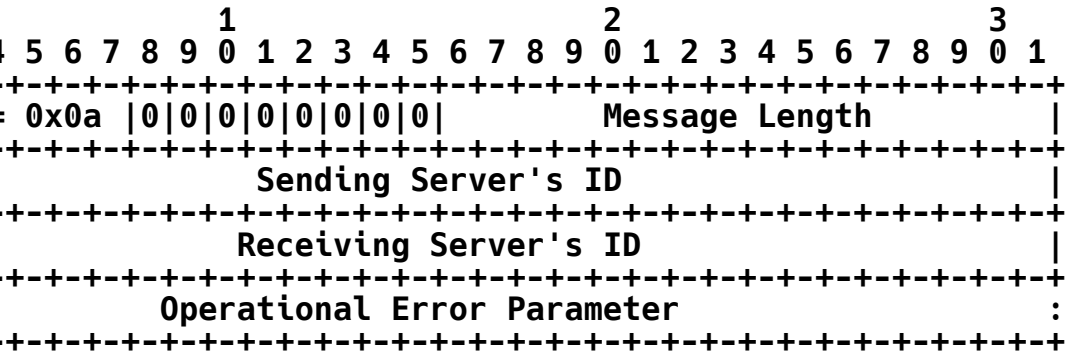
See Section 2.1.

Targeting Server's ID:

This is the ID of the peer ENRP that is the target of this takeover operation.

2.10. ENRP_ERROR Message

The ENRP_ERROR message is sent by a registrar to report an operational error to a peer ENRP server.



Sending Server's ID:

See Section 2.1.

Receiving Server's ID:

See Section 2.1.

Operational Error Parameter:

This parameter, defined in [RFC5354], indicates the type of error(s) being reported.

3. ENRP Operation Procedures

In this section, we discuss the operation procedures defined by ENRP. An ENRP server MUST follow these procedures when sending, receiving, or processing ENRP messages.

Many of the RSerPool events call for both server-to-server and PU/PE-to-server message exchanges. Only the message exchanges and activities between an ENRP server and its peer(s) are considered within the ENRP scope and are defined in this document.

Procedures for exchanging messages between a PE/PU and ENRP servers are defined in [RFC5352].

3.1. Methods for Communicating amongst ENRP Servers

Within an RSerPool operational scope, ENRP servers need to communicate with each other in order to exchange information, such as the pool membership changes, handlespace data synchronization, etc.

Two types of communications are used amongst ENRP servers:

- point-to-point message exchanges from one ENPR server to a specific peer server, and
- announcements from one server to all its peer servers in the operational scope.

Point-to-point communication is always carried out over an SCTP association between the sending server and the receiving server. Announcements are sent out via "group-casts" over the ENRP server channel.

3.2. ENRP Server Initialization

This section describes the steps a new ENRP server needs to take in order to join the other existing ENRP servers, or to initiate the handlespace service if it is the first ENRP server started in the operational scope.

3.2.1. Generate a Server Identifier

A new ENRP server MUST generate a non-zero, 32-bit server ID that is as unique as possible among all the ENRP servers in the operational scope, and this server ID MUST remain unchanged for the lifetime of the server. Normally, a good 32-bit random number will be good enough, as the server ID [RFC4086] provides some information on randomness guidelines.

Note, there is a very remote chance (about 1 in about 4 billion) that two ENRP servers in an operational scope will generate the same server ID and hence cause a server ID conflict in the pool. However, no severe consequence of such a conflict has been identified.

Note, the ENRP server ID space is separate from the PE Id space defined in [RFC5352].

3.2.2. Acquire Peer Server List

At startup, the ENRP server (the initiating server) will first attempt to learn of all existing peer ENRP servers in the same operational scope, or to determine that it is alone in the scope.

The initiating server uses an existing peer server to bootstrap itself into service. We call this peer server the mentor server.

3.2.2.1. Finding the Mentor Server

If the initiating server is told about one existing peer server through some administrative means (such as DNS query, configuration database, startup scripts, etc.), the initiating server MUST then use this peer server as its mentor server.

If multiple existing peer servers are specified, the initiating server MUST pick one of them as its mentor server and keep the others as its backup mentor servers.

If no existing peer server is specified, the initiating server MUST assume that it is alone in the operational scope, and MUST skip the procedures in Section 3.2.2.2 and Section 3.2.3 and MUST consider its initialization completed and start offering ENRP services.

3.2.2.2. Request Complete Server List from Mentor Peer

Once the initiating server finds its mentor peer server (by either discovery or administrative means), the initiating server MUST send an ENRP_LIST_REQUEST message to the mentor peer server to request a copy of the complete server list maintained by the mentor peer (see Section 3.4 for maintaining a server list).

The initiating server SHOULD start a MAX-TIME-NO-RESPONSE timer every time it finishes sending an ENRP_LIST_REQUEST message. If the timer expires before receiving a response from the mentor peer, the initiating server SHOULD abandon the interaction with the current mentor server and send a new server list request to a backup mentor peer, if one is available.

Upon the reception of this request, the mentor peer server SHOULD reply with an ENRP_LIST_RESPONSE message and include in the message body all existing ENRP servers known by the mentor peer.

Upon the reception of the ENRP_LIST_RESPONSE message from the mentor peer, the initiating server MUST use the server information carried in the message to initialize its own peer list.

However, if the mentor itself is in the process of startup and not ready to provide a peer server list (for example, the mentor peer is waiting for a response to its own ENRP_LIST_REQUEST to another server), it MUST reject the request by the initiating server and respond with an ENRP_LIST_RESPONSE message with the R flag set to '1', and with no server information included in the response.

In the case where its ENRP_LIST_REQUEST is rejected by the mentor peer, the initiating server SHOULD either wait for a few seconds and re-send the ENRP_LIST_REQUEST to the mentor server, or if there is a backup mentor peer available, select another mentor peer server and send the ENRP_LIST_REQUEST to the new mentor server.

3.2.3. Download ENRP Handlespace Data from Mentor Peer

After a peer list download is completed, the initiating server MUST request a copy of the current handlespace data from its mentor peer server, by taking the following steps:

1. The initiating server MUST first send an ENRP_HANDLE_TABLE_REQUEST message to the mentor peer, with the W flag set to '0', indicating that the entire handlespace is requested.

2. Upon the reception of this message, the mentor peer MUST start a download session in which a copy of the current handlespace data maintained by the mentor peer is sent to the initiating server in one or more ENRP_HANDLE_TABLE_RESPONSE messages. (Note, the mentor server may find it particularly desirable to use multiple ENRP_HANDLE_TABLE_RESPONSE messages to send the handlespace when the handlespace is large, especially when forming and sending out a single response containing a large handlespace may interrupt its other services.)

   If more than one ENRP_HANDLE_TABLE_RESPONSE message is used during the download, the mentor peer MUST use the M flag in each ENRP_HANDLE_TABLE_RESPONSE message to indicate whether this message is the last one for the download session. In particular, the mentor peer MUST set the M flag to '1' in the outbound ENRP_HANDLE_TABLE_RESPONSE if there is more data to be transferred and MUST keep track of the progress of the current download session. The mentor peer MUST set the M flag to '0' in the last ENRP_HANDLE_TABLE_RESPONSE for the download session and close the download session (i.e., removing any internal record of the session) after sending out the last message.

3. During the downloading, every time the initiating server receives an ENRP_HANDLE_TABLE_RESPONSE message, it MUST transfer the data entries carried in the message into its local handlespace database, and then check whether or not this message is the last one for the download session.

   If the M flag is set to '1' in the just processed ENRP_HANDLE_TABLE_RESPONSE message, the initiating server MUST send another ENRP_HANDLE_TABLE_REQUEST message to the mentor peer to request for the next ENRP_HANDLE_TABLE_RESPONSE message.

4. When unpacking the data entries from a ENRP_HANDLE_TABLE_RESPONSE message into its local handlespace database, the initiating server MUST handle each pool entry carried in the message using the following rules:

   A. If the pool does not exist in the local handlespace, the initiating server MUST create the pool in the local handlespace and add the PE(s) in the pool entry to the pool.

      When creating the pool, the initiation server MUST set the overall member selection policy type of the pool to the policy type indicated in the first PE.

   B. If the pool already exists in the local handlespace, but the PE(s) in the pool entry is not currently a member of the pool, the initiating server MUST add the PE(s) to the pool.

   C. If the pool already exists in the local handlespace AND the PE(s) in the pool entry is already a member of the pool, the initiating server SHOULD replace the attributes of the existing PE(s) with the new information. ENRP will make sure that the information stays up to date.

5. When the last ENRP_HANDLE_TABLE_RESPONSE message is received from the mentor peer and unpacked into the local handlespace, the initialization process is completed and the initiating server SHOULD start to provide ENRP services.

Under certain circumstances, the mentor peer itself may not be able to provide a handlespace download to the initiating server. For example, the mentor peer is in the middle of initializing its own handlespace database, or it currently has too many download sessions open to other servers.

In such a case, the mentor peer MUST reject the request by the initiating server and respond with an ENRP_HANDLE_TABLE_RESPONSE message with the R flag set to '1', and with no pool entries included in the response.

In the case where its ENRP_HANDLE_TABLE_REQUEST is rejected by the mentor peer, the initiating server SHOULD either wait for a few seconds and re-send the ENRP_HANDLE_TABLE_REQUEST to the mentor server, or if there is a backup mentor peer available, select another mentor peer server and send the ENRP_HANDLE_TABLE_REQUEST to the new mentor server.

A handlespace download session that has been started may get interrupted for some reason. To cope with this, the initiating server SHOULD start a timer every time it finishes sending an ENRP_HANDLE_TABLE_REQUEST to its mentor peer. If this timer expires without receiving a response from the mentor peer, the initiating server SHOULD abort the current download session and re-start a new handlespace download with a backup mentor peer, if one is available.

Similarly, after sending out an ENRP_HANDLE_TABLE_RESPONSE, and the mentor peer setting the M-bit to '1' to indicate that it has more data to send, it SHOULD start a session timer. If this timer expires without receiving another request from the initiating server, the mentor peer SHOULD abort the session, cleaning out any resource and record of the session.

3.3. Server Handlespace Update

This includes a set of update operations used by an ENRP server to inform its peers when its local handlespace is modified, e.g., addition of a new PE, removal of an existing PE, change of pool or PE properties.

3.3.1. Announcing Additions or Updates of PE

When a new PE is granted registration to the handlespace or an existing PE is granted a re-registration, the Home ENRP server uses this procedure to inform all its peers.

This is an ENRP announcement and is sent to all the peer of the Home ENRP server. See Section 3.1 on how announcements are sent.

An ENRP server MUST announce this update to all its peers in a ENRP_HANDLE_UPDATE message with the Update Action field set to 'ADD_PE', indicating the addition of a new PE or the modification of

an existing PE. The complete new information of the PE and the pool it belongs to MUST be indicated in the message with a PE parameter and a Pool Handle parameter, respectively.

The Home ENRP server SHOULD fill in its server ID in the Sending Server's ID field and leave the Receiving Server's ID blank (i.e., all 0s).

When a peer receives this ENRP_HANDLE_UPDATE message, it MUST take the following actions:

1. If the named pool indicated by the pool handle does not exist in its local copy of the handlespace, the peer MUST create the named pool in its local handlespace and add the PE to the pool as the first PE. It MUST then copy in all other attributes of the PE carried in the message.

   When the new pool is created, the overall member selection policy of the pool MUST be set to the policy type indicated by the PE.

2. If the named pool already exists in the peer's local copy of the handlespace *and* the PE does not exist, the peer MUST add the PE to the pool as a new PE and copy in all attributes of the PE carried in the message.

3. If the named pool exists *and* the PE is already a member of the pool, the peer MUST replace the attributes of the PE with the new information carried in the message.

3.3.2. Announcing Removal of PE

When an existing PE is granted de-registration or is removed from its handlespace for some other reasons (e.g., purging an unreachable PE, see Section 3.5 in [RFC5352]), the ENRP server MUST use this procedure to inform all its peers about the change just made.

This is an ENRP announcement and is sent to all the peers of the Home ENRP server. See Section 3.1 on how announcements are sent.

An ENRP server MUST announce the PE removal to all its peers in an ENRP_HANDLE_UPDATE message with the Update Action field set to DEL_PE, indicating the removal of an existing PE. The complete information of the PE and the pool it belongs to MUST be indicated in the message with a PE parameter and a Pool Handle parameter, respectively.

The sending server MUST fill in its server ID in the Sending Server's ID field and leave the Receiving Server's ID blank (i.e., set to all 0s).

When a peer receives this ENRP_HANDLE_UPDATE message, it MUST first find the pool and the PE in its own handlespace, and then remove the PE from its local handlespace. If the removed PE is the last one in the pool, the peer MUST also delete the pool from its local handlespace.

If the peer fails to find the PE or the pool in its handlespace, it SHOULD take no further actions.

3.4. Maintaining Peer List and Monitoring Peer Status

An ENRP server MUST keep an internal record on the status of each of its known peers. This record is referred to as the server's "peer list".

3.4.1. Discovering New Peer

If a message of any type is received from a previously unknown peer, the ENRP server MUST consider this peer a new peer in the operational scope and add it to the peer list.

The ENRP server MUST send an ENRP_PRESENCE message with the Reply-required flag set to '1' to the source address found in the arrived message. This will force the new peer to reply with its own ENRP_PRESENCE containing its full server information (see Section 2.1).

3.4.2. Server Sending Heartbeat

Every PEER-HEARTBEAT-CYCLE seconds, an ENRP server MUST announce its continued presence to all its peer with a ENRP_PRESENCE message. In the ENRP_PRESENCE message, the ENRP server MUST set the 'Replay_required' flag to '0', indicating that no response is required.

The arrival of this periodic ENRP_PRESENCE message will cause all its peers to update their internal variable "peer_last_heard" for the sending server (see Section 3.4.3 for more details).

3.4.3. Detecting Peer Server Failure

An ENRP server MUST keep an internal variable "peer_last_heard" for each of its known peers and the value of this variable MUST be updated to the current local time every time a message of any type (point-to-point or announcement) is received from the corresponding peer.

If a peer has not been heard for more than MAX-TIME-LAST-HEARD seconds, the ENRP server MUST immediately send a point-to-point ENRP_PRESENCE with the Reply_request flag set to '1' to that peer.

If the send fails or the peer does not reply after MAX-TIME-NO-RESPONSE seconds, the ENRP server MUST consider the peer server dead and SHOULD initiate the takeover procedure defined in Section 3.5.

3.5. Taking Over a Failed Peer Server

In the following descriptions, we call the ENRP server that detects the failed peer server and initiates the takeover the "initiating server" and the failed peer server the "target server". This allows the PE to continue to operate in case of a failure of their Home ENRP server.

3.5.1. Initiating Server Take-over Arbitration

The initiating server SHOULD first start the takeover arbitration process by sending an ENRP_INIT_TAKEOVER message to all its peer servers. See Section 3.1 on how announcements are sent. In the message, the initiating server MUST fill in the Sending Server's ID and Targeting Server's ID. The goal is that only one ENRP server takes over the PE from the target.

After announcing the ENRP_INIT_TAKEOVER message ("group-casting" to all known peers, including the target server), the initiating server SHOULD wait for an ENRP_INIT_TAKEOVER_ACK message from each of its known peers, except that of the target server.

Each peer receiving an ENRP_INIT_TAKEOVER message from the initiating server MUST take the following actions:

1. If the peer server determines that it (itself) is the target server indicated in the ENRP_INIT_TAKEOVER message, it MUST immediately announce an ENRP_PRESENCE message to all its peer ENRP servers in an attempt to stop this takeover process. This

indicates a false failure-detection case by the initiating server. The initiating server MUST stop the takeover operation by marking the target server as "active" and taking no further takeover actions.

2. If the peer server finds that it has already started its own takeover arbitration process on the same target server, it MUST perform the following arbitration:

   A. If the peer's server ID is smaller in value than the Sending Server's ID in the arrived ENRP_INIT_TAKEOVER message, the peer server MUST immediately abort its own take-over attempt by taking no further takeover actions of its own. Moreover, the peer MUST mark the target server as "not active" on its internal peer list so that its status will no longer be monitored by the peer, and reply to the initiating server with an ENRP_INIT_TAKEOVER_ACK message.

   B. Otherwise, the peer MUST ignore the ENRP_INIT_TAKEOVER message.

3. If the peer finds that it is neither the target server nor is in its own takeover process, the peer MUST: a) mark the target server as "not active" on its internal peer list so that its status will no longer be monitored by this peer, and b) MUST reply to the initiating server with an ENRP_INIT_TAKEOVER_ACK message.

Once the initiating server has received the ENRP_INIT_TAKEOVER_ACK message from all of its currently known peers (except for the target server), it MUST consider that it has won the arbitration and MUST proceed to complete the takeover, following the steps described in Section 3.5.2.

However, if it receives an ENRP_PRESENCE from the target server at any point in the arbitration process, the initiating server MUST immediately stop the takeover process and mark the status of the target server as "active".

3.5.2. Takeover Target Peer Server

The initiating ENRP server MUST first send, via an announcement, an ENRP_TAKEOVER_SERVER message to inform all its active peers that the takeover has been enforced. The target server's ID MUST be filled in the message. The initiating server SHOULD then remove the target server from its internal peer list.

Then, it SHOULD examine its local copy of the handlespace and claim ownership of each of the PEs originally owned by the target server, by following these steps:

1. mark itself as the Home ENRP server of each of the PEs originally owned by the target server;

2. send a point-to-point ASAP_ENDPOINT_KEEP_ALIVE message, with the 'H' flag set to '1', to each of the PEs. This will trigger the PE to adopt the initiating sever as its new Home ENRP server.

When a peer receives the ENRP_TAKEOVER_SERVER message from the initiating server, it SHOULD update its local peer list and PE cache by following these steps:

1. remove the target server from its internal peer list;

2. update the Home ENRP server of each PE in its local copy of the handlespace to be the sender of the message, i.e., the initiating server.

3.6. Handlespace Data Auditing and Re-synchronization

Message losses or certain temporary breaks in network connectivity may result in data inconsistency in the local handlespace copy of some of the ENRP servers in an operational scope. Therefore, each ENRP server in the operational scope SHOULD periodically verify that its local copy of handlespace data is still in sync with that of its peers.

This section defines the auditing and re-synchronization procedures for an ENRP server to maintain its handlespace data consistency.

3.6.1. Auditing Procedures

A checksum covering the data that should be the same is exchanged to figure out whether or not the data is the same.

The auditing of handlespace consistency is based on the following procedures:

1. An ENRP server SHOULD keep a separate PE checksum (a 16-bit integer internal variable) for each of its known peers and for itself. For an ENRP server with 'k' known peers, we denote these internal variables as "pe_checksum_pr0", "pe_checksum_pr1", ..., "pe_checksum_prk", where "pe_checksum_pr0" is the server's own PE checksum. The list of what these checksums cover and a detailed algorithm for calculating them is given in Section 3.6.2.

2. Each time an ENRP server sends out an ENRP_PRESENCE, it MUST include in the message its current PE checksum (i.e., "pe_checksum_pr0").

3. When an ENRP server (server A) receives a PE checksum (carried in an arrived ENRP_PRESENCE) from a peer ENRP server (server B), server A SHOULD compare the PE checksum found in the ENRP_PRESENCE with its own internal PE checksum of server B (i.e., "pe_checksum_prB").

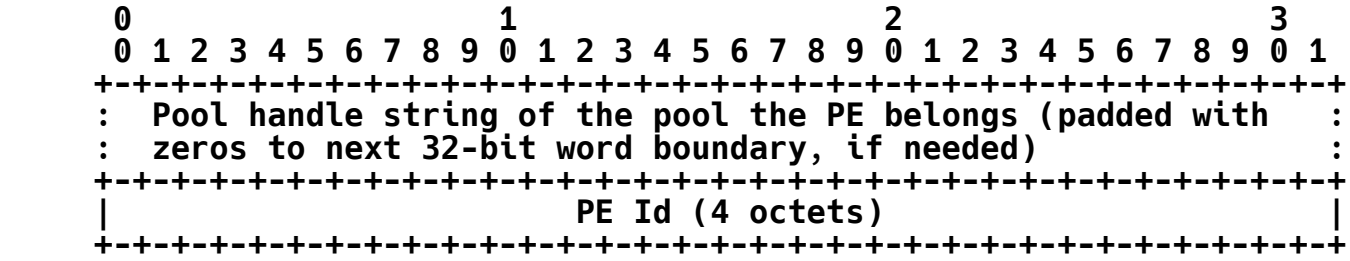
4. If the two values match, server A will consider that there is no handlespace inconsistency between itself and server B, and it should take no further actions.

5. If the two values do NOT match, server A SHOULD consider that there is a handlespace inconsistency between itself and server B, and a re-synchronization process SHOULD be carried out immediately with server B (see Section 3.6.3).

3.6.2. PE Checksum Calculation Algorithm

When an ENRP server (server A) calculates an internal PE checksum for a peer (server B), it MUST use the following algorithm.

Let us assume that in server A's internal handlespace, there are currently 'M' PEs that are owned by server B. Each of the 'M' PEs will then contribute to the checksum calculation with the following byte block:



Note, these are not TLVs. This byte block gives each PE a unique byte pattern in the scope. The 16-bit PE checksum for server B "pe_checksum_prB" is then calculated over the byte blocks contributed by the 'M' PEs one by one. The PE checksum calculation MUST use the Internet algorithm described in [RFC1071].

Server A MUST calculate its own PE checksum (i.e., "pe_checksum_pr0") in the same fashion, using the byte blocks of all the PEs owned by itself.

Note, whenever an ENRP finds that its internal handlespace has changed (e.g., due to PE registration/de-registration, receiving peer updates, removing failed PEs, downloading handlespace pieces from a peer, etc.), it MUST immediately update all its internal PE checksums that are affected by the change.

Implementation Note: when the internal handlespace changes (e.g., a new PE added or an existing PE removed), an implementation need not re-calculate the affected PE checksum; it can instead simply update the checksum by adding or subtracting the byte block of the corresponding PE from the previous checksum value.

3.6.3. Re-Synchronization Procedures

If an ENRP server determines that there is inconsistency between its local handlespace data and a peer's handlespace data with regard to the PEs owned by that peer, it MUST perform the following steps to re-synchronize the data:

1. The ENRP server SHOULD first "mark" every PE it knows about that is owned by the peer in its local handlespace database;

2. The ENRP server SHOULD then send an ENRP_HANDLE_TABLE_REQUEST message with the W flag set to '1' to the peer to request a complete list of PEs owned by the peer;

3. Upon reception of the ENRP_HANDLE_TABLE_REQUEST message with the W flag set to '1', the peer server SHOULD immediately respond with an ENRP_HANDLE_TABLE_RESPONSE message listing all PEs currently owned by the peer.

4. Upon reception of the ENRP_HANDLE_TABLE_RESPONSE message, the ENRP server SHOULD transfer the PE entries carried in the message into its local handlespace database. If a PE entry being transferred already exists in its local database, the ENRP server MUST replace the entry with the copy found in the message and remove the "mark" from the entry.

5. After transferring all the PE entries from the received ENRP_HANDLE_TABLE_RESPONSE message into its local database, the ENRP server SHOULD check whether there are still PE entries that remain "marked" in its local handlespace. If so, the ENRP server SHOULD silently remove those "marked" entries.

Note, similar to what is described in Section 3.2.3, the peer may reject the ENRP_HANDLE_TABLE_REQUEST or use more than one ENRP_HANDLE_TABLE_RESPONSE message to respond.

3.7. Handling Unrecognized Messages or Unrecognized Parameters

When an ENRP server receives an ENRP 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 Sections 3 and 4 in [RFC5354].

According to the rules, if an error report to the message sender is needed, the ENRP server that discovered the error SHOULD send back an ENRP_ERROR message with a proper error cause code.

4. Variables and Thresholds

4.1. Variables

peer_last_heard - The local time that a peer server was last heard (via receiving either a group-cast or point-to-point message from the peer).

pe_checksum_pr - The internal 16-bit PE checksum that an ENRP server keeps for a peer. A separate PE checksum is kept for each of its known peers as well as for itself.

4.2. Thresholds

PEER-HEARTBEAT-CYCLE - The period for an ENRP server to announce a heartbeat message to all its known peers. (Default=30 secs.)

MAX-TIME-LAST-HEARD - Pre-set threshold for how long an ENRP server will wait before considering a silent peer server potentially dead. (Default=61 secs.)

MAX-TIME-NO-RESPONSE - Pre-set threshold for how long a message sender will wait for a response after sending out a message. (Default=5 secs.)

5. IANA Considerations

This document (RFC 5353) is the reference for all registrations described in this section. All registrations have been listed on the RSerPool Parameters page.

5.1. A New Table for ENRP Message Types

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

Type	Message Name	Reference
0x00	(Reserved by IETF)	RFC 5353
0x01	ENRP_PRESENCE	RFC 5353
0x02	ENRP_HANDLE_TABLE_REQUEST	RFC 5353
0x03	ENRP_HANDLE_TABLE_RESPONSE	RFC 5353
0x04	ENRP_HANDLE_UPDATE	RFC 5353
0x05	ENRP_LIST_REQUEST	RFC 5353
0x06	ENRP_LIST_RESPONSE	RFC 5353
0x07	ENRP_INIT_TAKEOVER	RFC 5353
0x08	ENRP_INIT_TAKEOVER_ACK	RFC 5353
0x09	ENRP_TAKEOVER_SERVER	RFC 5353
0x0a	ENRP_ERROR	RFC 5353
0x0b-0xff	(Available for assignment)	RFC 5353

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

5.2. A New Table for Update Action Types

Update Types are maintained by IANA. Two initial values have been assigned by IANA. IANA created a new table, "Update Action Types":

Type	Update Action	Reference
0x0000	ADD_PE	RFC 5353
0x0001	DEL_PE	RFC 5353
0x0002-0xffff	(Available for assignment)	RFC 5353

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

5.3. Port Numbers

The references for the already assigned port numbers

enrp-udp 9901/udp

enrp-sctp 9901/sctp

enrp-sctp-tls 9902/sctp

have been updated to RFC 5353.

5.4. SCTP Payload Protocol Identifier

The reference for the already assigned ENRP payload protocol identifier 12 have been updated to RFC 5353.

6. Security Considerations

We present a summary of the threats to the RSerPool architecture and describe security requirements in response 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.

6.1. 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, 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)
```

6.2. 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. 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 RFC 791 [RFC0791] and RFC 2460 [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 of RFC 5352) 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.

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.

6.3. 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 the 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.

7. Acknowledgments

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

8. References

8.1. Normative References

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

[RFC1071] Braden, R., Borman, D., Partridge, C., and W. Plummer, "Computing the Internet checksum", RFC 1071, September 1988.

[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.

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

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

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

[RFC5352] Stewart, R., Xie, Q., Stillman, M., and M. Tuexen, "Aggregate Server Access Protocol (ASAP)", RFC 5352, 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.

8.2. Informative References

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

[SCTPSOCKET] Stewart, R., Poon, K., Tuexen, M., Yasevich, V., and P. Lei, "Sockets API Extensions for Stream Control Transmission Protocol (SCTP)", Work in Progress, July 2008.

Authors' Addresses

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

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

Phone:
EMail: randall@lakerest.net

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

Phone:
EMail: maureen.stillman@nokia.com

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

EMail: tuexen@fh-muenster.de

Aron J. Silverton
Sun Microsystems, Inc.
10 S. Wacker Drive
Suite 2000
Chicago, IL 60606
USA

Phone:
EMail: ajs.ietf@gmail.com

Full Copyright Statement

Intellectual Property