Level 3 multihoming shim protocol
draft-ietf-shim6-proto-02.txt

Status of this Memo

Copyright Notice

Abstract

The SHIM6 working group is exploring a layer 3 shim approach for providing locator agility below the transport protocols, so that multihoming can be provided for IPv6 with failover and load spreading properties, without assuming that a multihomed site will have a provider independent IPv6 address prefix which is announced in the global IPv6 routing table. The hosts in a site which has multiple provider allocated IPv6 address prefixes, will use the shim6 protocol specified in this document to setup state with peer hosts, so that the state can later be used to failover to a different locator pair,

should the original one stop working.

This document picks a particular approach to such a protocol and tries to flush out a bunch of details, with the hope that the WG can better understand the details in this proposal as well as discovering and understanding alternative designs that might be better. Thus this proposal is my no means cast in stone as the direction; quite to the contrary it is a depth first exploration of the design space.

Table of Contents

1. Introduction

The SHIM6 working group, and the MULTI6 WG that preceded it, is exploring a layer 3 shim approach for providing locator agility below the transport protocols, so that multihoming can be provided for IPv6 with failover and load spreading properties [14], without assuming that a multihomed site will have a provider independent IPv6 address which is announced in the global IPv6 routing table. The hosts in a site which has multiple provider allocated IPv6 address prefixes, will use the shim6 protocol specified in this document to setup state with peer hosts, so that the state can later be used to failover to a different locator pair, should the original one stop working.

This document takes the outlines contained in [22] and [21] and expands to an actual proposed protocol.

We assume that redirection attacks are prevented using the mechanism specified in HBA [6].

The WG mailing list is discussing the scheme used for reachability detection [7]. The schemes that are being discussed are Context Unreachability Detection (CUD) or Force Bidirectional communication Detection (FBD). This document doesn't discuss the tradeoffs between the two, but it does suggest a set of keepalive and probe messages that are sufficient to handle both. Once the WG has decided which approach to take, we can remove the unneeded messages.

There is a related but slightly separate issue of how the hosts can find which of the locator pairs is working after a failure. This is discussed in [8].

NOTE that the direction taken in the latest version of [8] is to use FBD and some new SHIM6 message types. Some of that work has been reflected in this document, but there are other edits that remain.

1.1 Goals

The goals for this approach is to:

- o Preserve established communications through failures, for example, TCP connections and application communications using UDP.
- o Have no impact on upper layer protocols in general and on transport protocols in particular.
- o Address the security threats in [17] through a separate document [6], and techniques described in this document.
- o No extra roundtrip for setup; deferred setup.
- o Take advantage of multiple locators/addresses for load spreading so that different sets of communication to a host (e.g., different connections) might use different locators of the host.

1.2 Non-Goals

The assumption is that the problem we are trying to solve is site multihoming, with the ability to have the set of site locator prefixes change over time due to site renumbering. Further, we assume that such changes to the set of locator prefixes can be relatively slow and managed; slow enough to allow updates to the DNS to propagate. But it is not a goal to try to make communication survive a renumbering event (which causes all the locators of a host to change to a new set of locators). This proposal does not attempt to solve, perhaps related, problems such as host multihoming or host mobility.

This proposal also does not try to provide an IP identifier. Even though such a concept would be useful to ULPs and applications, especially if the management burden for such a name space was zero and there was an efficient yet secure mechanism to map from identifiers to locators, such a name space isn't necessary (and furthermore doesn't seem to help) to solve the multihoming problem.

1.3 Locators as Upper-layer Identifiers

Central to this approach is to not introduce a new identifier name space but instead use one of the locators as the upper-layer ID, while allowing the locators used in the address fields to change over time in response to failures of using the original locator.

This implies that the ULID selection is performed as today's default address selection as specified in [12]. Underneath, and transparently, the multihoming shim selects working locator pairs with the initial locator pair being the ULID pair. When communication fails the shim can test and select alternate locators. A subsequent section discusses the issues when the selected ULID is not initially working hence there is a need to switch locators up front.

Using one of the locators as the ULID has certain benefits for applications which have long-lived session state, or performs callbacks or referrals, because both the FQDN and the 128-bit ULID work as handles for the applications. However, using a single 128-bit ULID doesn't provide seamless communication when that locator is unreachable. See [18] for further discussion of the application implications.

There has been some discussion of using non-routable locators, such as unique-local addresses [16], as ULIDs in a multihoming solution. While this document doesn't specify all aspects of this, it is believed that the approach can be extended to handle such a case.

For example, the protocol already needs to handle ULIDs that are not initially reachable. Thus the same mechanism can handle ULIDs that are permanently unreachable from outside their site. The issue becomes how to make the protocol perform well when the ULID is not reachable, for instance, avoiding any timeout and retries in this case. In addition one would need to understand how the ULAs would be entered in the DNS to avoid a performance impact on existing, non-shim6 aware, IPv6 hosts potentially trying to communicate to the (unreachable) ULA.

1.4 IP Multicast

IP Multicast requires that the IP source address field contain a topologically correct locator for interface that is used to send the packet, since IP multicast routing uses both the source address and the destination group to determine where to forward the packet. (This isn't much different than the situation with widely implemented ingress filtering [10] for unicast.)

While in theory it would be possible to apply the shim re-mapping of the IP address fields between ULIDs and locators, the fact that all the multicast receivers would need to know the mapping to perform, makes such an approach difficult in practice. Thus it makes sense to have multicast ULPs operate directly on locators and not use the shim. This is quite a natural fit for protocols which use RTP [13], since RTP already has an explicit identifier in the form of the SSRC field in the RTP headers. Thus the actual IP address fields are not important to the application.

1.5 Renumbering Implications

As stated above, this approach does not try to make communication survive renumbering. However, the fact that a ULID might be used with a different locator over time open up the possibility that communication between two ULIDs might continue to work after one or both of those ULIDs are no longer reachable as locators, for example due to a renumbering event. This opens up the possibility that the ULID (or at least the prefix on which it is based) is reassigned to another site while it is still being used (with another locator) for existing communication.

Worst case we could end up with two separate hosts using the same ULID while both of them are communicating with the same host.

This potential source for confusion can be avoided if we require that any communication using a ULID must be terminated when the ULID becomes invalid (due to the underlying prefix becoming invalid).

However, this might be an overkill. Even when an IPv6 prefix is retired and reassigned to some other site, there is still a very small probability that another host in that site picks the same 128 bit address (whether using DHCPv6, stateless address autoconfiguration, or picking a random interface ID [11]). Should the identical address be used by another host, then there still wouldn't be a problem until that host attempts to communicate with the same host with which the initial user of the IPv6 address was communicating.

1.6 Placement of the shim

```
                        -----------------------
                        | Transport Protocols |
                        -----------------------

          ------ ------- -------------- -------------     IP endpoint
          | AH | | ESP | | Frag/reass | | Dest opts |     sub-layer
          ------ ------- -------------- -------------

                     ---------------------
                     | shim6 shim layer |
                     ---------------------

                            ------                       IP routing
                            | IP |                       sub-layer
                            ------
```

Figure 1: Protocol stack

The proposal uses an multihoming shim layer within the IP layer, i.e., below the ULPs, as shown in Figure 1, in order to provide ULP independence. The multihoming shim layer behaves as if it is associated with an extension header, which would be placed after any routing-related headers in the packet (such as any hop-by-hop options, or routing header). However, when the locator pair is the ULID pair there is no data that needs to be carried in an extension header, thus none is needed in that case.

Layering AH and ESP above the multihoming shim means that IPsec can be made to be unaware of locator changes the same way that transport protocols can be unaware. Thus the IPsec security associations remain stable even though the locators are changing. The MOBIKE WG is looking at ways to have IPsec security associations survive even though the IP addresses changes, which is a different approach.

Layering the fragmentation header above the multihoming shim makes reassembly robust in the case that there is broken multi-path routing

which results in using different paths, hence potentially different source locators, for different fragments. Thus, effectively the multihoming shim layer is placed between the IP endpoint sublayer, which handles fragmentation, reassembly, and IPsec, and the IP routing sublayer, which selects which next hop and interface to use for sending out packets.

Applications and upper layer protocols use ULIDs which the shim6 layer will map to/from different locators. The shim6 layer maintains state, called host-pair context, per ULID pairs (that is, applies to all ULP connections between the ULID pair) in order to perform this mapping. The mapping is performed consistently at the sender and the receiver, thus from the perspective of the upper layer protocols, packets appear to be sent using ULIDs from end to end, even though the packets travel through the network containing locators in the IP address fields, and even though those locators might be changed by the transmitting shim6 layer.

The context state in this approach is maintained per remote ULID i.e. approximately per peer host, and not at any finer granularity. In particular, it is independent of the ULPs and any ULP connections. However, the forking capability enables shim-aware ULPs to use more than one locator pair at a time for an single ULID pair.

```
----------------------------          ----------------------------
| Sender A                 |          | Receiver B               |
|                          |          |                          |
|     ULP                  |          |     ULP                  |
|      | src ULID(A)=L1(A) |          |      ^                   |
|      | dst ULID(B)=L1(B) |          |      | src ULID(A)=L1(A) |
|      v                   |          |      | dst ULID(B)=L1(B) |
|   multihoming shim       |          |   multihoming shim       |
|      | src L2(A)         |          |      ^                   |
|      | dst L3(B)         |          |      | src L2(A)         |
|      v                   |          |      | dst L3(B)         |
|      IP                  |          |      IP                  |
----------------------------          ----------------------------
       |                                     ^
       ------- cloud with some routers -------
```

Figure 2: Mapping with changed locators

The result of this consistent mapping is that there is no impact on the ULPs. In particular, there is no impact on pseudo-header checksums and connection identification.

Conceptually one could view this approach as if both ULIDs and locators are being present in every packet, but with a header

compression mechanism applied that removes the need for the ULIDs once the state has been established. In order for the receiver to recreate a packet with the correct ULIDs there might be a need to include some "compression tag" in the data packets. This would serve to indicate the correct context to use for decompression when the locator pair in the packet is insufficient to uniquely identify the context.

2. Terminology

This document uses the terms MUST, SHOULD, RECOMMENDED, MAY, SHOULD NOT and MUST NOT defined in RFC 2119 [1]. The terms defined in RFC 2460 [2] are also used.

2.1 Definitions

This document introduces the following terms (taken from [22]):

upper layer protocol (ULP)	A protocol layer immediately above IP. Examples are transport protocols such as TCP and UDP, control protocols such as ICMP, routing protocols such as OSPF, and internet or lower-layer protocols being "tunneled" over (i.e., encapsulated in) IP such as IPX, AppleTalk, or IP itself.
interface	A node's attachment to a link.
address	An IP layer name that contains both topological significance and acts as a unique identifier for an interface. 128 bits. This document only uses the "address" term in the case where it isn't specific whether it is a locator or an identifier.
locator	An IP layer topological name for an interface or a set of interfaces. 128 bits. The locators are carried in the IP address fields as the packets traverse the network.
identifier	An IP layer name for an IP layer endpoint (stack name in [24]). The transport endpoint name is a function of the transport protocol and would typically include the IP identifier plus a port number. NOTE: This proposal does not specify any new form of IP layer identifier, but still separates the identifying and locating properties of the IP

	addresses.
upper-layer identifier (ULID)	An IP address which has been selected for communication with a peer to be used by the upper layer protocol. 128 bits. This is used for pseudo-header checksum computation and connection identification in the ULP. Different sets of communication to a host (e.g., different connections) might use different ULIDs in order to enable load spreading. Since the ULID is just one of the IP locators/ addresses of the node, there is no need for a separate name space and allocation mechanisms.
address field	The source and destination address fields in the IPv6 header. As IPv6 is currently specified this fields carry "addresses". If identifiers and locators are separated these fields will contain locators for packets on the wire.
FQDN	Fully Qualified Domain Name
Host-pair context	The state that the multihoming shim maintains. The context is for a ULID pair, and is identified by a context tag for each direction of the communication.
Context tag	Each end of the context allocates a context tag for the context. This is used to uniquely associate both received control packets and payload packets with the shim6 Payload extension header as belonging to the context.
Current locator pair	Each end of the context has a current locator pair which is used to send packets to be peer. The two ends might use different current locator pairs though.
Default context	At the sending end, the shim uses the ULID pair (passed down from the ULP) to find the context for that pair. Thus, normally, a host can have at most one context for a ULID pair. We call this the "default context".

Context forking	A mechanism which allows ULPs that are aware of multiple locators to use separate contexts for the same ULID pair, in order to be able use different locator pairs for different communication to the same ULID. Context forking causes more than just the default context to be created for a ULID pair.

2.2 Notational Conventions

A, B, and C are hosts. X is a potentially malicious host.

FQDN(A) is the domain name for A.

Ls(A) is the locator set for A, which consists of the locators L1(A), L2(A), ... Ln(A).

ULID(A) is an upper-layer ID for A. In this proposal, ULID(A) is always one member of A's locator set.

This document also makes use of internal conceptual variables to describe protocol behavior and external variables that an implementation must allow system administrators to change. The specific variable names, how their values change, and how their settings influence protocol behavior are provided to demonstrate protocol behavior. An implementation is not required to have them in the exact form described here, so long as its external behavior is consistent with that described in this document. See Section 6 for a description of the conceptual data structures.

3. Assumptions

The general approach of a level3 shim as well as this specific proposal makes the following assumptions:

- When there is ingress filtering in the ISPs, that the use of all <source, destination> locator pairs will cause the packets to exit using different ISPs so that all exit ISPs can be tried. Since there might be only one destination locator, when the peer supports shim6 but is not multihomed, this implies that the selection of the exit ISP should be related to the source address in the packets.
- Even without ingress filtering, there is the assumption that if the host tries all <source, destination> locator pairs, that it has done a good enough job of trying to find a working path to the peer. Since we want the protocol to provide benefits even if the peer has a single locator, this seems to imply that the choice of source locator needs to somehow affect the exit path from the

site.

4. Protocol Overview

The shim6 protocol operates in several phases over time. The following sequence illustrates the concepts:

- An application on host A decides to contact B using some upper-layer protocol. This results in the ULP on A sending packets to B. We call this the initial contact. Assuming the IP addresses selected by Default Address Selection [12] work, then there is no action by the shim at this point in time. Any shim context establishment can be deferred until later.
- Some heuristic on A or B (or both) determine that it might make sense to make this communication robust against locator failures. For instance, this heuristic might be that more than 50 packets have been sent or received. This makes the shim initiate the 4-way context establishment exchange.

  As a result of this exchange, both A and B will know a list of locators for each other.

  If the context establishment exchange fails, the initiator will then know that the other end does not support shim6, and will revert to standard unicast behavior for the session.
- Communication continues without any change for the ULP packets. In addition, there might be some messages exchanged between the shim sub-layers for (un)reachability detection.
- At some point in time something fails. Depending on the approach to reachability detection, there might be some advise from the ULP, or the shim (un)reachability detection might discover that there is a problem.

  At this point in time one or both ends of the communication need to probe and explore the different alternate locator pairs until a working pair is found, and rehome to using that pair.
- Once a working alternative locator pair has been found, the shim will rewrite the packets on transmit, and tag the packets with shim6 Payload message as an extension header, which contains the receiver's context tag. The receiver will use the <Source Locator, Destination Locator, Context Tag> to find the context state which will indicate which addresses to place in the IPv6 header before passing the packet up to the ULP. The result is that from the perspective of the ULP the packet passes unmodified end-to-end, even though the IP routing infrastructure sends the packet to a different locator.
- The shim (un)reachability detection will monitor the new locator pair as it monitored the original locator pair, so that subsequent failures can be detected.

- In addition to failures detected based on end-to-end observations, one endpoint might be know for certain that one or more of its locators is not working. For instance, the network interface might have failed or gone down (at layer 2), or an IPv6 address might have become invalid. In such cases the host can signal its peer that this address is no longer recommended to try. Thus this triggers something similar to a failure handling in that a new, working locator pair must be found.

  The Working Group has discussed whether or not hosts can express other forms of locator preferences. If this is the case, a change in the preferences can be signaled to the peer, which might make the peer choose to try a different locator pair. Thus, this can also be treated similarly to a failure.
- When the shim thinks that the context state is no longer used, it can garbage collect the state; there is no coordination necessary with the peer host before the state is removed. There is an error message defined to be able to signal when there is no context state, which can be used to detect and recover from both premature garbage collection, as well as complete state loss (crash and reboot) of a peer.

The ULP packets in shim6 are carried completely unmodified as long as the ULID pair is used as the locator pair. After a switch to a different locator pair the packets are "tagged" with a shim6 extension header, so that the receiver can always determine the context to which they belong. This is accomplished by including an 8-octet "shim payload" extension header before the (extension) headers that are processed by the IP endpoint sublayer and ULPs.

4.1 Context Tags

A context between two hosts is actually a context between two ULIDs. The context is identified by a pair of context tags. Each end gets to allocate a context tag, and once the context is established, the shim6 control messages contain the context tag that the receiver of the message allocated. Thus at a minimum the combination of <peer ULID, local ULID, local context tag> MUST uniquely identify one context.

In addition, the non-shim6 messages, which we call payload packets, will not contain the ULIDs after a failure. This introduces the requirement that the <peer locator, local locator, local context tag> MUST uniquely identify the context. Since the peer's set of locators might be dynamic the simplest form of unique allocation of the local context tag is to pick a number that is unique on the host. Hosts which serve multiple ULIDs using disjoint sets of locators can maintain the context tag allocation per such disjoint set.

The mechanism for detecting a loss of context state at the peer that is currently proposed in this document assumes that the receiver can tell the packets that need locator rewriting, even after it has lost all state (e.g., due to a crash followed by a reboot). This is achieved because after a rehoming event the packets that need receive-side rewriting, carry the Payload Message extension header.

Even though we do not overload the flow label field to carry the context tag, any protocol (such as RSVP or NSIS) which signals information about flows from the host stack to devices in the path, need to be made aware of the locator agility introduced by a layer 3 shim, so that the signaling can be performed for the locator pairs that are currently being used.

TBD: add forking - multiple contexts between ULID pairs, default context, etc. Need to explain that context forking assumes an API from the ULP.

TBD: add that shim can be disabled for some ULP traffic if we define an API for this purpose.

4.2 Securing shim6

The mechanisms are secured using a combination of techniques:

- o The HBA technique [6] for validating the locators to prevent an attacker from redirecting the packet stream to somewhere else.
- o Requiring a Reachability Probe+Reply before a new locator is used as the destination, in order to prevent 3rd party flooding attacks.
- o The first message does not create any state on the responder. Essentially a 3-way exchange is required before the responder creates any state. This means that a state-based DoS attack (trying to use up all of memory on the responder) at least provides an IPv6 address that the attacker was using.
- o The context establishment messages use nonces to prevent replay attacks.

4.3 Overview of Shim Control Messages

The shim context establishment is accomplished using four messages; I1, R1, I2, R2. Normally they are sent in that order from initiator and responder, respectively. Should both ends attempt to set up context state at the same time (for the same ULID pair), then their I1 messages might cross in flight, and result in an immediate R2 message. [The names of these messages are borrowed from HIP [23].]

There is a No Context error message defined, when a control or payload packet arrives and there is no matching context state at the

receiver. When such a message is received, it will result in the destruction of the shim context and a re-establishment.

The peers' lists of locators are normally exchanged as part of the context establishment exchange. But the set of locators might be dynamic. For this reason there is a Locator List Update message and acknowledgement.

Even though the list of locators is fixed, a host might determine that some preferences might have changed. For instance, it might determine that there is a locally visible failure that implies that some locator(s) are no longer usable. Currently this mechanism has a separate message pair (Rehome Request and acknowledgement), but perhaps this can be encoded using the Locator List Update message pair with a preference option and no change to the list of locators.

At least two approaches (CUD and FBD) have been discussed for the shim (un)reachability detection [7]. This document attempt to define messages for both cases; once the WG has picked an approach we can delete any unneeded messages.

The CUD approach uses a probe message and acknowledgement, which can be suppressed e.g. using positive advise from the ULP. This message pair also seems needed to verify that the host is indeed present at a new locator before the data stream is redirected to that locator, in order to prevent 3rd party DoS attacks.

The FBD approach uses a keepalive message, which is sent when a host has received packets from the peer, but the ULP has not given the host an opportunity to send any payload packet to the peer.

The above probe and keepalive messages assume we have an established host-pair context. However, communication might fail during the initial context (that is, when the application or transport protocol is trying to setup some communication). If we want the shim to be able to optimize discovering a working locator pair in that case, we need a mechanism to test the reachability of locators independent of some context. We define a locator pair test message and acknowledgement for this purpose, even though it isn't yet clear whether we need such a thing.

Finally, when the context is established and there is a failure there needs to be a way to probe and explore the set of locator pairs to efficiently find a working pair. We define an explore message as a place holder for some mechanism in this space [8].

4.4 Locator Validation

Before a host can use a locator (different than the ULID) as the source locator, it must know that the peer will accept packets with that source locator as being part of this context. The peer might wish to do some verification of the locator before accepting it as a source address. This document does not require any such verification. But if it is done by a host, in all cases such verification need to be finished before the host acknowledges the new locator, by sending an Update Acknowledgement message, R2 an message.

Before a host can use a locator (different than the ULID) as the destination locator it must perform the full verification of the locator. This includes both verifying it using HBA/CGA, and verifying that the ULID is indeed reachable at the locator. The latter in order to prevent 3rd party flooding attacks.

5. Message Formats

The shim6 messages are all carried using a new IP protocol number TBD [to be assigned by IANA]. The shim6 messages have a common header, defined below, with some fixed fields, followed by type specific fields.

The shim6 messages are structured as an IPv6 extension header since the Payload Message is used to carry the ULP packets after a locator switch. The shim6 control messages use the same extension header formats so that a single "protocol number" needs to be allowed through firewalls in order for shim6 to function across the firewall.

5.1 Common shim6 Message Format

The first 17 bits of the shim6 header is common for the Payload Message and the control messages and looks as follows:

```
 0                   1
 0 1 2 3 4 5 6 7 8 9 0 1 2 3 4 5 6
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |P|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Next Header:	The payload which follows this header.
Hdr Ext Len:	8-bit unsigned integer. Length of the shim6 header in 8-octet units, not including the first 8 octets.

```
   P:             A single bit to distinguish Payload messages from
                  control messages.
```

5.2 Payload Message Format

The payload message is used to carry ULP packets where the receiver must replace the content of the source and or destination fields in the IPv6 header before passing the packet to the ULP. Thus this extension header is included when the locators pair that is used is not the same as the ULID pair.

Since the shim is placed between the IP endpoint sub-layer and the IP routing sub-layer in the host, the shim header will be placed before any endpoint extension headers (fragmentation headers, destination options header, AH, ESP), but after any routing related headers (hop-by-hop extensions header, routing header, a destinations options header which precedes a routing header). When tunneling is used, whether IP-in-IP tunneling or the special form of tunneling that Mobile IPv6 uses (with Home Address Options and Routing header type 2), there is a choice whether the shim applies inside the tunnel or outside the tunnel, which effects the location of the shim6 header.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Next Header  |       0       |1|       Reserved              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Receiver Context Tag                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Fields:
   Next Header:   The payload which follows this header.
   Hdr Ext Len:   0 (since the header is 8 octets).
   P:             Set to one.  A single bit to distinguish this from the
                  shim6 control messages.
   Reserved:      Reserved for future use.  Zero on transmit.  MUST be
                  ignored on receipt.
   Receiver Context Tag: 32-bit unsigned integer.  Allocated by the
                  receiver for use to identify the context (together
                  with the source and destination locators).
```

5.3 Common Shim6 Control header

The common part of the header has a next header and header extension length field which is consistent with the other IPv6 extension headers, even if the next header value is always "NO NEXT HEADER" for the control messages; only the payload messages use the Next Header field.

The shim6 headers must be a multiple of 8 octets, hence the minimum size is 8 octets.

The common message header is as follows:

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|  Next Header  |  Hdr Ext Len  |0|     Type    |Type specific|0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Checksum           |                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+                               |
|                    Type specific format                       |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Next Header:	8-bit selector. Normally set to NO_NXT_HDR (59). Indicates the next header value for the shim6 payload messages.
Hdr Ext Len:	8-bit unsigned integer. Length of the shim6 header in 8-octet units, not including the first 8 octets.
P:	Set to zero. A single bit to distinguish this from the shim6 payload messages.
Type:	7-bit unsigned integer. Identifies the actual message from the table below.
0:	A single bit (set to zero) which allows shim6 and HIP to have a common header format yet telling shim6 and HIP messages apart.
Checksum:	16-bit unsigned integer. The checksum is the 16-bit one's complement of the one's complement sum of the entire shim6 header message starting with the shim6 next header field, and ending as indicated by the Hdr Ext Len. Thus when there is a payload following the shim6 header, the payload is NOT included in the shim6 checksum.

Type Value	Message
1	I1 (first establishment message from the initiator)
2	R1 (first establishment message from the responder)
3	I2 (2nd establishment message from the initiator)
4	R2 (2nd establishment message from the responder)
5	No Context Error
6	Update Request
7	Update Acknowledgement
8	Reachability Probe
9	Reachability Reply
10	Keepalive
11	SHIM6 Probe Message

Table 1

5.4 I1 Message Format

The I1 message is the first message in the context establishment exchange.

```
 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
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|       59      |  Hdr Ext Len  |0|  Type = 1   |   Reserved1 |0|
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|            Checksum           |           Reserved2           |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                  Initiator Context Tag                        |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                   Initiator Nonce                             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
+                            Options                            +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Next Header:	NO_NXT_HDR (59).
Type:	1
Reserved1:	7-bit field. Reserved for future use. Zero on transmit. MUST be ignored on receipt.

```
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Initiator Context Tag: 32-bit field.  The Context Tag the initiator
                  has allocated for the context.
   Initiator Nonce: 32-bit unsigned integer.  A random number picked by
                  the initiator which the responder will return in the
                  R1 message.

   The following options are allowed in the message:
   ULID pair:     TBD Do we need to carry the ULIDs, or assume they are
                  the same as the address fields in the IPv6 header?
                  Depends on how we handle failures during initial
                  contact.  We also need it to be able to reestablish
                  the host-pair context after a failure when one end has
                  lost the context state.
```

5.5 R1 Message Format

```
   The R1 message is the second message in the context establishment
   exchange.  The responder sends this in response to an I1 message,
   without creating any state specific to the initiator.

    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 2   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Initiator Nonce                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                       Responder Nonce                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                            Options                            +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Fields:
   Next Header:   NO_NXT_HDR (59).
   Type:          2
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
```

```
   Initiator Nonce: 32-bit unsigned integer.  Copied from the I1
                  message.
   Responder Nonce: 32-bit unsigned integer.  A number picked by the
                  responder which the initiator will return in the I2
                  message.

   The following options are allowed in the message:
   Responder Validator: Variable length option.  Typically a hash
                  generated by the responder, which the responder uses
                  together with the Responder Nonce value to verify that
                  an I2 message is indeed sent in response to a R1
                  message, and that the parameters in the I2 message are
                  the same as those in the I1 message.
```

5.6 I2 Message Format

```
   The I2 message is the third message in the context establishment
   exchange.  The initiator sends this in response to a R1 message,
   after checking the Initiator Nonce, etc.

    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 3   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Initiator Context Tag                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Initiator Nonce                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Responder Nonce                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                         Options                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Fields:
   Next Header:   NO_NXT_HDR (59).
   Type:          3
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
```

```
   Initiator Context Tag: 32-bit field.  The Context Tag the initiator
                  has allocated for the context.
   Initiator Nonce: 32-bit unsigned integer.  A random number picked by
                  the initiator which the responder will return in the
                  R2 message.
   Responder Nonce: 32-bit unsigned integer.  Copied from the R1
                  message.

   The following options are allowed in the message:
   Responder Validator: Variable length option.  Just a copy of the
                  Validator option in the R1 message.
   ULID pair:     TBD Do we need to carry the ULIDs, or assume they are
                  the same as the address fields in the IPv6 header?  We
                  also need it to be able to reestablish the host-pair
                  context after a failure when one end has lost the
                  context state.
   Locator list:  Optionally sent when the initiator immediately wants
                  to tell the responder its list of locators.  When it
                  is sent, the necessary HBA/CGA information for
                  validating the locator list MUST also be included.
   Locator Preferences: Optionally sent when the locators don't all have
                  equal preference.
   CGA Parameter Data Structure: Included when the locator list is
                  included so the receiver can verify the locator list.
   CGA Signature: Included when the some of the locators in the list use
                  CGA (and not HBA) for validation.
```

5.7 R2 Message Format

The R2 message is the fourth message in the context establishment exchange. The responder sends this in response to an I2 message. The R2 message is also used when both hosts send I1 messages at the same time and the I1 messages cross in flight.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 4   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                  Responder Context Tag                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Initiator Nonce                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                          Options                              +
   |                                                               |
```

```
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Fields:
   Next Header:   NO_NXT_HDR (59).
   Type:          4
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Responder Context Tag: 32-bit field.  The Context Tag the responder
                  has allocated for the context.
   Initiator Nonce: 32-bit unsigned integer.  Copied from the I2
                  message.

   The following options are allowed in the message:
   Locator List:  Optionally sent when the responder immediately wants
                  to tell the initiator its list of locators.  When it
                  is sent, the necessary HBA/CGA information for
                  validating the locator list MUST also be included.
   Locator Preferences: Optionally sent when the locators don't all have
                  equal preference.
   CGA Parameter Data Structure: Included when the locator list is
                  included so the receiver can verify the locator list.
   CGA Signature: Included when the some of the locators in the list use
                  CGA (and not HBA) for validation.
```

5.8 No Context Error Message Format

Should a host receive a packet with a shim Payload message or shim6 control message, such a a locator update, and the host does not have any context state for the locators (in the IPv6 source and destination fields) and the context tag, then it will generate a No Context Error. The error includes the packet that was received, subject to the packet not exceeding 1280 octets.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 5   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                           Options                             +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

```
   Next Header:   NO_NXT_HDR (59).
   Type:          5
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
```

The following options are allowed in the message:

```
   Packet in Error: Variable length option containing the IPv6 packet
                  that was in error, starting with the IPv6 header, and
                  normally containing the full packet.  If the resulting
                  No Context Error message would exceed 1280 octets, the
                  Packet In Error option will not include the full
                  packet in error in order to limit the error to 1280
                  octets.
```

5.9 Update Request Message Format

The Update Request Message is used to update either the list or locators, the locator preferences, and both. When the list of locators is updated, the message also contains the option(s) necessary for HBA/CGA to secure this. The basic sanity check that prevents off-path attackers from generating bogus updates is the context tag in the message.

The update message contains options (the Locator List and the Locator Preferences) that, when included, completely replace the previous locator list and locator preferences, respectively. Thus there is no mechanisms to just send deltas to the locator list.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 6   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Receiver Context Tag                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                    Request Nonce                              |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                           Options                             +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

```
   Next Header:   NO_NXT_HDR (59).
   Type:          6
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Receiver Context Tag: 32-bit field.  The Context Tag the receiver has
                  allocated for the context.
   Request Nonce: 32-bit unsigned integer.  A random number picked by
                  the initiator which the peer will return in the
                  acknowledgement message.

   The following options are allowed in the message:
   Locator List:  The list of the senders (new) locators.  The locators
                  might be unchanged and only the preferences have
                  changed.
   Locator Preferences: Optionally sent when the locators don't all have
                  equal preference.
   CGA Signature: Included when the some of the locators in the list use
                  CGA (and not HBA) for validation.
```

5.10 Update Acknowledgement Message Format

This message is sent in response to a Update Request message. It implies that the Update Request has been received, and that any new locators in the Update Request can now be used as the source locators of packets. But it does not imply that the (new) locators have been verified to be used as a destination, since the host might defer the verification of a locator until it sees a need to use a locator as the destination.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 7   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Receiver Context Tag                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Request Nonce                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                         Options                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

```
   Next Header:   NO_NXT_HDR (59).
   Type:          7
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Receiver Context Tag: 32-bit field.  The Context Tag the receiver has
                  allocated for the context.
   Request Nonce: 32-bit unsigned integer.  Copied from the Update
                  Request message.
```

No options are currently defined for this message.

5.11 Reachability Probe Message Format

TBD: Given [8] we do not need this message any more.

The Reachability Probe message is used to prevent 3rd party DoS attacks, and can also be used to verify whether a context is reachable at a given locator should that be needed for the general reachability detection mechanism (e.g., if we pick the CUD mechanism where one end sends probes and expects a reply).

Before a host uses a locator for the peer that is different than the ULID, it needs to verify that the peer is indeed present at that locator by sending a Context Verify and receiving an acknowledgement. This message includes the ULID pair as well as the context tag, so that the peer can indeed verify that it has that ULID and that the context tag is correct.

```
    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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 8   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Receiver Context Tag                      |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                        Request Nonce                          |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                            Options                            +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

```
   Next Header:   NO_NXT_HDR (59).
   Type:          8
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Receiver Context Tag: 32-bit field.  The Context Tag the receiver has
                  allocated for the context.
   Request Nonce: 32-bit unsigned integer.  A random number picked by
                  the initiator which the responder will return in the
                  acknowledgement message.
```

The following options are allowed in the message:

```
   ULID pair:     The ULID pair that is being probed.
```

5.12 Reachability Reply Message Format

TBD: Given [8] we do not need this message any more.

This is sent in response to a Reachability Probe message. Although, if the receiver of the Reachability Probe does not have a matching context it will send a No Context Error 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 9   |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Receiver Context Tag                     |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                         Request Nonce                         |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                            Options                            +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

```
   Fields:
   Next Header:   NO_NXT_HDR (59).
   Type:          9
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
```

```
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Receiver Context Tag: 32-bit field.  The Context Tag the receiver has
                  allocated for the context.
   Request Nonce: 32-bit unsigned integer.  Copied from the request
                  message.

   The following options are allowed in the message:
   ULID pair:     The ULID pair that is being probed.  Copied from the
                  Probe message.
```

5.13 Keepalive Message Format

```
   TBD: Given [8] we do not need this message any more.

   The keepalive message would be used if we decide to do the Force
   Bidirectional communication as a way to get verification that the
   locator pair continues to work.  If we are not going to do FBD we
   probably will not need this 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
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |       59      |  Hdr Ext Len  |0|  Type = 10  |   Reserved1 |0|
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |            Checksum           |           Reserved2           |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                   Receiver Context Tag                        |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                      Request Nonce                            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                                                               |
   +                         Options                               +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+

   Fields:
   Next Header:   NO_NXT_HDR (59).
   Type:          10
   Reserved1:     7-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Reserved2:     16-bit field.  Reserved for future use.  Zero on
                  transmit.  MUST be ignored on receipt.
   Receiver Context Tag: 32-bit field.  The Context Tag the receiver has
                  allocated for the context.
```

Request Nonce: 32-bit unsigned integer. Copied from the Reachability
               Probe message.

No options are currently defined for this message.

5.14 SHIM6 Probe Message Format

This message and its semantics are defined in [8]. The idea behind that mechanism is to be able to handle the case when one locator pair works in from A to B, and another locator pair works from B to A, but there is no locator pair which works in both directions. The protocol mechanism is that as A is sending probe messages to B, B will observe which locator pairs it has received from and report that back in probe messages it is sending to A.

5.15 Option Formats

All of the TLV parameters have a length (including Type and Length fields) which is a multiple of 8 bytes. When needed, padding MUST be added to the end of the parameter so that the total length becomes a multiple of 8 bytes. This rule ensures proper alignment of data. If padding is added, the Length field MUST NOT include the padding. Any added padding bytes MUST be zeroed by the sender, and their values SHOULD NOT be checked by the receiver.

Consequently, the Length field indicates the length of the Contents field (in bytes). The total length of the TLV parameter (including Type, Length, Contents, and Padding) is related to the Length field according to the following formula:

Total Length = 11 + Length - (Length + 3) % 8;

```
 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            |C|             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|                                                               |
/                          Contents                             /
/                                               +-+-+-+-+-+-+-+-+
|                                               |    Padding    |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Type:	15-bit identifier of the type of option. The options defined in this document are below.
C:	Critical. One if this parameter is critical, and MUST be recognized by the recipient, zero otherwise. An implementation might view the C bit as part of the Type field, by multiplying the type values in this specification by two.
Length:	Length of the Contents, in bytes.
Contents:	Parameter specific, defined by Type.
Padding:	Padding, 0-7 bytes, added if needed.

Option Name	Type
Validator	1
Locator List	2
Locator Preferences	3
CGA Parameter Data Structure	4
CGA Signature	5
ULID Pair	6
Packet In Error	7
SHIM6 Event Option	8

Table 2

5.15.1 Validator Option Format

The responder can choose exactly what input uses to compute the validator, and what one-way function (MD5, SHA1) it uses, as long as the responder can verify that the validator it receives back in the I2 message is indeed one that 1) it computed, 2) it computed for the particular context, and 3) that it isn't a replayed I2 message.

One way for the responder to do this is to maintain a single secret (S) and a running counter for the Responder Nonce. For each I1 message, the responder can then increase the counter, use the counter value as the responder nonce, and use the following information as input to the one-way function:

- o The the secret S
- o That Responder Nonce
- o The Initiator Context Tag from the I1 message
- o The ULIDs from the I1 message
- o The locators from the I1 message (strictly only needed if they are different from the ULIDs)

```
     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 = 1            |0|            Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ~                           Validator                           ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Validator: Variable length content whose interpretation is local
               to the responder.

5.15.2 **Locator List Option Format**

The Locator List Option is used to carry all the locators of the sender. Note that the order of the locators is important, since the Locator Preferences refers to the locators by using the index in the list.

Note that we carry all the locators in this option even though some of them can be created automatically from the CGA Parameter Data Structure.

TBD: We can get a simpler format if we split this into two options: one with the locators and one with just the verification methods.

```
     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 = 2            |0|            Length           |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                     Locator List Generation                   |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |  Num Locators |            N Octets of Verification Method    |
    +-+-+-+-+-+-+-+-+                                               |
    ~                                                               ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ~                     Locators 1 through N                      ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Locator List Generation: 32-bit unsigned integer. Indicates a
               generation number which is increased by one for each
               new locator list. This is used to ensure that the
               index in the Locator Preferences refer to the right
               version of the locator list.

Num Locators:	8-bit unsigned integer. The number of locators that are included in the option. We call this number "N" below.
Verification Method:	N octets. The i'th octet specifies the verification method for the i'th locator.
Locators:	N 128-bit locators.

The defined verification methods are:

Value	Method
0	Reserved
1	HBA
2	CGA
3-255	Reserved

Table 3

5.15.3 Locator Preferences Option Format

The Locator Preferences option can have some flags to indicate whether or not a locator is known to work. In addition, the sender can include a notion of preferences. It might make sense to define "preferences" as a combination of priority and weight the same way that DNS SRV records has such information. The priority would provide a way to rank the locators, and within a given priority, the weight would provide a way to do some load sharing. See [9] for how SRV defines the interaction of priority and weight.

The minimum notion of preferences we need is to be able to indicate that a locator is "dead". We can handle this using a single octet flag for each locator.

We can extend that by carrying a larger "element" for each locator. This document presently also defines 2-octet and 3-octet elements, and we can add more information by having even larger elements if need be.

The locators are not included in the preference list. Instead, the first element refers to locator that was in the first element in the Locator List option. The generation number carried in this option and the Locator List option is used to verify that they refer to the same version of the locator list.

```
    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 = 3          |0|            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |                     Locator List Generation                   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |  Element Len  |  Element[1]   |  Element[2]   |  Element[3]   |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                              ...                              ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Case of Element Len = 1 is depicted.

Fields:

Locator List Generation:	32-bit unsigned integer. Indicates a generation number for the locator list to which the elements should apply.
Element Len:	8-bit unsigned integer. The length in octets of each element. This draft defines the cases when the length is 1, 2, or 3.
Element[i]:	A field with a number of octets defined by the Element Len field. Provides preferences for the i'th locator in the Locator List option that is in use.

When the Element length equals one, then the element consists of only a flags field. The set of flags is TBD: Assume there will be two initially: BROKEN and TEMPORARY. The intent of the latter is to allow the distinction between more stable addresses and less stable addresses when shim6 is combined with IP mobility, when we might have more stable home locators, and less stable care-of-locators.

When the Element length equals two, the the element consists of a 1 octet flags field followed by a 1 octet priority field. The priority has the same semantics as the priority in DNS SRV records.

When the Element length equals three, the the element consists of a 1 octet flags field followed by a 1 octet priority field, and a 1 octet weight field. The weight has the same semantics as the weight in DNS SRV records.

5.15.4 CGA Parameter Data Structure Option Format

This option contains the CGA parameter data structure (hereafter called the PDS). When HBA is used to validate the locators, the PDS contains the HBA multiprefix extension. When CGA is used to validate the locators, in addition to the CGA PDS, the signature will need to be included as a CGA Signature option.

```
    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 = 4          |0|            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                   CGA Parameter Data Structure                ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:
CGA Parameter Data Structure: Variable length content. Content
               defined in [5] and [6].

5.15.5 CGA Signature Option Format

When CGA is used for validation of one or more of the locators in the PDS, then the message in question will need to contain this option.

```
    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 = 5          |0|            Length             |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   ~                        CGA Signature                          ~
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:
CGA Signature: A variable-length field containing a PKCS#1 v1.5
               signature, constructed by using the sender's private
               key over the following sequence of octets:

1. The 128-bit CGA Message Type tag [CGA] value for SHIM6, 0x4A 30 5662 4858 574B 3655 416F 506A 6D48. (The tag value has been generated randomly by the editor of this specification.).
2. The Locator List Generation value of the correspondent Locator List Option.
3. The subset of locators included in the correspondent Locator List Option which validation method is set to CGA. The locators MUST be included in the order they are listed in the Locator List Option.

5.15.6 ULID Pair Option Format

It isn't clear whether we need this option. It depends whether we want to be able to setup a context for a ULID pair when that ULID pair can't be used to communicate. Thus the IPv6 addresses in the context establishment would not be the ULIDs.

```
     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 = 2          |0|            Length             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                          Sender ULID                          +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    |                                                               |
    +                         Receiver ULID                         +
    |                                                               |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Reserved:	48-bit field. Reserved for future use. Zero on transmit. MUST be ignored on receipt.
Sender ULID:	A 128-bit IPv6 address.
Receiver ULID:	A 128-bit IPv6 address.

5.15.7 Packet In Error Option Format

```
     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 = 7          |0|            Length             |
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
    ~           IPv6 header, shim6/TCP/UDP header, etc              ~
    +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Fields:

Packet:	A variable length field which contains the packet in error starting with the IPv6 header.

5.15.8 SHIM6 Event Option Format

This option is defined in [8].

6. Conceptual Model of a Host

This section describes a conceptual model of one possible data structure organization that hosts will maintain for the purposes of shim6. The described organization is provided to facilitate the explanation of how the shim6 protocol should behave. This document does not mandate that implementations adhere to this model as long as their external behavior is consistent with that described in this document.

6.1 Conceptual Data Structures

The key conceptual data structure for the shim6 protocol is the host pair context. This is a data structure which contains the following information:

- o The peer ULID; ULID(peer)
- o The local ULID; ULID(local)
- o The list of peer locators, with their preferences; Ls(peer)
- o For each peer locator, the validation method to use (from the Locator List option).
- o For each peer locator, a bit whether it has been validated using HBA or CGA, and a bit whether the locator has been probed to verify that the ULID is present at that location.
- o The preferred peer locator - used as destination; Lp(peer)
- o The set of local locators and the preferences; Ls(local)
- o The preferred local locator - used as source; Lp(local)
- o The context tag used to transmit control messages and ULP packets - allocated by the peer; CT(peer)
- o The context to expect in received control messages and extension headers - allocated by the local host; CT(local)
- o Reachability state for the locator pairs.
- o During pair exploration, information about the probe messages that have been sent and received.

The receiver finds the context by looking it up using <Source Locator, Destination Locator, CT(local)>, where the context tag is in the shim header. The sender needs to be able to find the context state when a ULP packet is passed down from the ULP. In that case the lookup key is the pair of ULIDs.

7. Establishing Host Pair Contexts

Host pair contexts are established using a 4-way exchange, which allows the responder to avoid creating state on the first packet. As part of this exchange each end allocates a context tag, and it shares this context tag and its set of locators with the peer.

In some cases the 4-way exchange is not necessary, for instance when both ends try to setup the context at the same time, or when recovering from a context that has been garbage collected or lost at one of the hosts.

7.1 Normal context establishment

The normal context establishment consists of a 4 message exchange in the order of I1, R1, I2, R2.

```
        Initiator                             Responder

              ------------- I1 -------------->

              <------------ R1 ---------------

              ------------- I2 -------------->

              <------------ R2 ---------------
```

Figure 24

7.2 Concurrent context establishment

When both ends try to initiate a context for the same ULID pair, then we might end up with crossing I1 messages, or since the no state is created when receiving the I1, a host might send a I1 after having sent a R1 message.

Since a host remembers that it has sent an I1, it can respond to an I1 from the peer (for the same ULID), with a R2.

```
        Initiator                             Responder

              -\
                ---\
                    ---\                  /---
                        --- I1 ---\   /---
                                   ---\
                       /--- I1 ---/    ---\
                  /---                     -->
              <---

              -\
                ---\
                    ---\                  /---
                        --- R2 ---\   /---
                                   ---\
                       /--- R2 ---/    ---\
                  /---                     -->
              <---
```

Figure 25

If a host has received an I1 and sent an R1, then a ULP can trigger it to send an I1 message itself, since it doesn't retain any state when receiving the I1 message. Thus while one end is sending an I1

the other is sending an I2.

```
        Initiator                          Responder

              -\
                ---\
                    ---\
                        --- I1 ---\
                                   ---\
                                       ---\
                                           -->

                                          /---
                                      /---
                                   ---
                       /--- R1--/
                  /---
              <---

              -\
                ---\
                    ---\                  /---
                        --- I2---\   /---
                                   ---\
                       /--- I1 ---/    ---\
                  /---                     -->
              <---

              -\
                ---\
                    ---\                  /---
                        --- R2 ---\   /---
                                   ---\
                       /--- R2 ---/    ---\
                  /---                     -->
              <---
```

Figure 26

7.3 Context recovery

Due to garbage collection, we can end up with one end having and using the context state, and the other end not having any state. We need to be able to recover this state at the end that has lost it, before we can use it.

This need can arise in two cases:

- The communication is working using the ULID pair as the locator pair, but a problem arises, and the end that has retained the context state decides to probe and explore alternate locator pairs.
- The communication is working using a locator pair that is not the ULID pair, hence the ULP packets sent from a peer that has retained the context state use the shim payload header.

In both cases the result is that the peer without state receives a shim message for which it has to context for the <source locator, destination locator, context tag>.

In both of those case we can recover the context by having the node which doesn't have a context state, send back an R1bis [TBD] message, and have this complete a recover with a I2 and R2 message.

If one end has garbage collected or lost the context state, it might try to create the context state (for the same ULID pair), by sending an I1 message. The peer can simply reply with an R2 message in this case.

7.4 Context confusion

Since each end might garbage collect the context state we can have the case when one end has retained the context state and tries to use it, while the other end has lost the state. We discussed this in the previous section on recovery. But for the same reasons, when one host retains context tag X for ULID pair <A1, B1>, the other end might end up allocating that context tag for another ULID pair, e.g., <A3, B1> between the same hosts. In this case we can not use the recovery mechanisms since there needs to be separate context tags for the two ULID pairs.

This type of "confusion" can be observed in two cases (assuming it is A that has retained the state and B has dropped it):

- B decides to create a context for ULID pair <A3, B1>, and allocates X as its context tag for this, and sends an I1 to A.
- A decides to create a context for ULID pair <A3, B1>, and starts the exchange by sending I1 to B. When B receives the I2 message, it allocates X as the context tag for this context.

In both cases, A can detect that B has allocated X for ULID pair <A3, B1> even though that A still X as CT(peer) for ULID pair <A1, B1>. Thus A can detect that B must have lost the context for <A1, B1>.

The solution to this issue is TBD. The know possibilities are:

- Have A forcibly destroy the context for <A1, B1>, so that it can accept the new context for <A3, B1>.

- o Have A accept the context for <A3, B1>, forget about the old context, but initiate a new (replacement) context for <A1, B1> by sending an I1 message. That I1 through R2 exchange will make B allocate a new context tag for <A1, B1>.
- o Avoid the problem by changing the context tag allocation so that A and B allocates half of the bits (16 each) of the context tags, so that even if one end looses state, the peer can make sure that the context tags for each context are unique.

7.5 Sending I1 messages

When the shim layer decides to setup a context for a ULID pair, it starts by allocating and initializing the context state for its end. As part of this it assigns its context tag to the context. Then it can send an I1 message.

If the host does not receive an I2 or R2 message in response to the I1 message, then it needs to retransmit the I1 message. The retransmissions should use a retransmission timer with binary exponential backoff to avoid creating congestion issues for the network when lots of hosts perform this.

If, after several retransmissions, there is no response, then most likely the peer does not implement the shim6 protocol, or there could be a firewall that blocks the protocol. In this case it makes sense for the host to remember to not try again to establish a host pair context with that ULID. However, any such negative caching should retained for a limit time; a few minutes would be appropriate, to allow things to recover should the host not be reachable at all when the shim tries to establish the context.

If the host receives an ICMP error with "payload type unknown" and the included packet is the I1 packet it just sent, then this is a more reliable indication that the peer ULID does not implement shim6.

7.6 Receiving I1 messages

If the host looks up a context for the ULID pair and the peer's (not its) context tag. If it finds such a context, the it needs to verify that the locators in the message are in fact part of the locator sets that are recorded in the existing context state. If this is not the case, then the I1 message MUST be silently ignored. (This can only happen when there is an ULID pair option in the I1 message.) If the locators are ok, then the host can respond with an R2 message as if it had received an I2 message and not an I1 message.

If there is no existing context state, then the host forms a verifier and sends this back to the peer in an I2 message. No state is

created on the host in this case.

7.7 Receiving R1 messages

When the host receives an R1 message, it verifies that the nonce matches what it sent in the I1 message, and that it has context state for the ULID pair. It then sends an I2 message, which includes the verifier option that was in the R1 message. The I2 message also includes A's locator list and the CGA parameter data structure. If CGA (and not HBA) is used to verify the locator list, then A also signs the key parts of the message and includes a CGA signature option containing the signature.

The host may receive an R1[bis] TBD message that was not sent in response to an I1 message but instead sent as a result of context recovery. The difference between an R1bis and an R1 message is that the former use the context tag of the responder. TBD how there are handled and whether they are identical to an R1.

7.8 Retransmitting I2 messages

If the initiator does not receive an R2 message after sending an I2 message it MAY retransmit the I2 message. But since the verifier option might have a limited lifetime, that is, the peer might reject verifier options that are too old to avoid replay attacks, the initiator SHOULD fall back to retransmitting the I1 message when there is no response to one or a few I2 messages.

7.9 Receiving I2 messages

The responder checks that the nonce and the verifier option is consistent with what it might have sent in a recent R1 message (by verifying the hash it computed.) If this is ok, then the host checks if it already has context state for the ULID pair and the CT(peer). If it has such state, the I2 message was probably a retransmission. In this case the host sends an R2 message.

If there is no context state, the responder allocates a context tag (CT(local)) and creates the context state for the context. It records the peer's locator set as well as its own locator set in the context. It MAY verify the peers locator set at this point in time, but the requirement is that a locator MUST be verified before the host starts sending packets to that locator, thus the host MAY defer the verification until later.

The host forms an R2 message with its locators and its context tag, and includes the necessary options so that the peer can verify the locators.

R2 messages are never retransmitted. If the R2 message is lost, then the initiator will retransmit either the I2 or I1 message. Either retransmission will cause the responder to find the context state and respond with an R2 message.

7.10 Receiving R2 messages

The initiator can receive an R2 message in response to either an I1 or an I2 message, but the handling of the R2 is the same in both cases. The host first verifies that the nonce is the same as the one it sent (in the I1 or I2 message). If it doesn't match, the R2 message is silently dropped.

Then the host records the information from the R2 message in the context state. It records the peer's locator set in the context. It MAY verify the peers locator set at this point in time, but the requirement is that a locator MUST be verified before the host starts sending packets to that locator, thus the host MAY defer the verification until later.

8. No Such Content Errors

TBD

The Interim Meeting discussed ways to recover the context state at one end when the other end sees a failure (and starts sending Probe messages). The discussed approach is to use a R1 (or R1bis) message in response to a message with an unknown context, which would cause the context to be recreated.

The idea is that on receipt of a SHIM6 payload packet where there is no current SHIM6 context at the receiver, the receiver is to respond with an R1bis packet in order to re-establish SHIM6 context. The R1bis packet differs from the R1 packet in that an R1 packet echoes the I1 fields, while this R1bis offers state back to the sender. One key difference is that the I1 packet contains the initiator's context tag, while the payload message header contains the receivers context tag. Either way the next control packet is an I2 in response. The senders previous context state is to be flushed in receipt of the R2 packet following the R1bis, I2 exchange.

The details of this type of exchange needs to be worked out, but the likely result is that we will not need a separate "No context" error message.

9. Handling ICMP Error Messages

The routers in the path as well as the destination might generate

various ICMP error messages, such as host unreachable, packet too big, and payload type unknown. It is critical that these packets make it back up to the ULPs so that they can take appropriate action.

When the ULP packets are sent unmodified, that is, while the initial locators=ULIDs are working, this introduces no new concerns; an implementation's existing mechanism for delivering these errors to the ULP will work. But when the shim on the transmitting side replaces the ULIDs in the IP address fields with some other locators, then an ICMP error coming back will have a "packet in error" which is not a packet that the ULP sent. Thus the implementation will have to apply the reverse mapping to the "packet in error" before passing the ICMP error up to the ULP.

This mapping is different than when receiving ULP packets from the peer, because in that case the packets contain CT(local). But the ICMP errors have a "packet in error" with CT(peer) since they were intended to be received by the peer. In any case, since the <Source Locator, Destination Locator, CT(peer)> has to be unique when received by the peer, the local host should also only be able to find one context that matches this tuple.

If the ULP packet had been encapsulated in a shim6 payload message, then this extension header must be removed. The result needs to be that the ULP receives an ICMP error where the contained "packet in error" looks as if the shim did not exist.

10. Teardown of the Host Pair Context

Each host can unilaterally decide when to tear down a host-pair context. It is RECOMMENDED that hosts not tear down the context when they know that there is some upper layer protocol that might use the context. For example, an implementation might know this is there is an open socket which is connected to the ULID(peer). However, there might be cases when the knowledge is not readily available to the shim layer, for instance for UDP applications which not not connect their sockets, or any application which retains some higher level state across (TCP) connections and UDP packets.

Thus it is RECOMMENDED that implementations minimize premature teardown by observing the amount of traffic that is sent and received using the context, and only after it appears quiescent, tear down the state.

TBD: The Interim meeting discussed whether it was feasible to relax this so that one can end up with an asymmetric distribution of the context state and still get (most of) the shim benefits. For example, the busy server would go through the context setup but would

quickly remove the context state after this (in order to save memory) but the not-so-busy client would retain the context state. The context recover mechanism presented in Section 7.3 would then be recreate the state should the client send either a shim control message (e.g., probe message because it sees a problem), or a ULP packet in an payload extension header (because it had earlier failed over to an alternative locator pair, but had been silent for a while). This seems to provide the benefits of the shim as long as the client can detect the failure. If the client doesn't send anything, and it is the server that tries to send, then it will not be able to recover because the shim on the server has no context state, hence doesn't know any alternate locator pairs.

11. Updating the Locator Pairs

TBD

The validation issues for the locators carried in the Locator Update message are specified in Section 4.4.

12. Various Probe Mechanisms

TBD

13. Rehoming to a Different Locator Pair

TBD

14. Sending ULP Payloads

When there is no context state for the ULID pair on the sender, there is no effect on how ULP packets are sent. If the host is using some heuristic for determining when to perform a deferred context establishment, then the host might need to do some accounting (count the number of packets sent and received) even before there is a host-pair context.

If there is a host-pair context for the ULID pair, then the sender needs to verify whether context uses the ULIDs as locators, that is, whether Lp(peer) == ULID(peer) and Lp(local) == ULID(local).

If this is the case, then packets will be sent unmodified by the shim. If it is not the case, then the logic in Section 14.1 will need to be used.

There will also be some maintenance activity relating to (un)reachability detection, whether packets are sent with the original locators or not. The details of this is out of scope for

this document and will be covered is follow-ons to [7].

14.1 Sending ULP Payload after a Switch

When sending packets, if there is a host-pair context for the ULID pair, and the ULID pair is no longer used as the locator pair, then the sender needs to transform the packet. Apart from replacing the IPv6 source and destination fields with a locator pair, an 8-octet header is added so that the receiver can find the context and inverse the transformation.

First, the IP address fields are replaced. The IPv6 source address field is set to Lp(local) and the destination address field is set to Lp(peer). NOTE that this MUST NOT cause any recalculation of the ULP checksums, since the ULP checksums are carried end-to-end and the ULP pseudo-header contains the ULIDs which are preserved end-to-end.

The sender skips any "routing sub-layer extension headers" that the ULP might have included, thus it skips any hop-by-hop extension header, any routing header, and any destination options header that is followed by a routing header. After any such headers the shim6 extension header will be added. This might be before a Fragment header, a Destination Options header, an ESP or AH header, or a ULP header.

The inserted shim6 Payload extension header includes the peer's context tag.

15. Receiving Packets

As in normal IPv6 receive side packet processing the receiver parses the (extension) headers in order. Should it find a shim6 extension header it will look at the type field in that header. If the type is Payload message, then the packet must be passed to the shim6 payload handling for rewriting. (Otherwise, the shim6 control messages are handled as specified in other parts of this document.)

The receiver extracts the context tag from the payload message header, and uses this together with the IPv6 source and destination address fields to find a host-pair context. If no context is found, the receiver SHOULD generate a No Such Context error message (see Section 8).

With the context in hand, the receiver can now replace the IP address fields with the ULIDs kept in the context. Finally, the Payload extension header is removed from the packet (so that the ULP doesn't get confused by it), and the next header value in the preceding header is set to be the actual protocol number for the payload. Then

the packet can be passed to the protocol identified by the next header value (which might be some function associated with the IP endpoint sublayer, or a ULP).

If the host is using some heuristic for determining when to perform a deferred context establishment, then the host might need to do some accounting (count the number of packets sent and received) for packets that does not have a shim6 extension header. But the need for this depends on what heuristics the implementation has chosen.

16. Initial Contact

TBD Describe what inital contact is (basically some non-shim communication starts between two ULIDs), and what the implications are of failures. Basic option is to rely on the application retrying and RFC 3484bis ordering of source and destination ULIDs.

17. Open Issues

The following open issues are known:

- o Forking the context state. On the mailing list we've discussed the need to fork the context state, so that different ULP streams can be sent using different locator pairs. No protocol extensions are needed if any forking is done independently by each endpoint. But if we want A to be able to tell B that certain traffic (a 5-tuple?) should be forked, then we need a way to convey this in the shim6 protocol. The hard part would be defining what selectors can be specified for the filter which determines which traffic uses which of the forks. So the question is whether we really need signaling for forking, or whether it is sufficient to allow each endpoint to do its own selection of which locator pair it is using for which traffic.
- o If we allow forking, it seems like the mechanism for reachability detection, whether it is CUD or FBD, must be applied separately for each locator pair that is in use. Without forking a single locator pair will be in use for each host-pair context, hence things would be simpler.
- o What happens when a host runs out of N bit context tags? When is it safe for a host to reuse a context tag? With the unilateral teardown one end might discard the context state long before the other end.
- o Should a host explicitly fail communication when a ULID becomes invalid (based on RFC 2462 lifetimes or DHCPv6), or should we let the communication continue using the invalidated ULID (it can certainly work since other locators will be used).
- o Should we rename "host-pair context" to be "ULID-pair context"? If we've decided this is per ULID pair that might make sense.

- o We need to pick some initial retransmit timers for I1 and I2. Is 4 seconds ok?
- o Should we require that the R1 verifier be usable for some minimum time so that the initiator knows for how long time it can safely retransmit I2 before it needs to go back to sending I1 again?
- o Should we expand the context tag from 32 to 47 bits?
- o Should we make the receiver not use the source locator to find the context, but instead only use the context tag? (and optionally, the destination locator). This would provide some flexibility for the future. The potential downside, which we would need to understand, is packet injection. *If* there is ingress filtering, then we get some extra checking by including the source locator in the lookup. But an on-path attacker can inject packets at will, whether the source locator is part of the lookup or not. An off-path attacker would have a hard time to guess a 47-bit number.
- o Include locator list in R1 message to deal with R2 being dropped?
- o Should we allow a host to intentionally discard the context state, with the assumption that the peer is responsible to maintain it, and detect failures? This might be useful in asymetric case, e.g. a server which serves lots of clients, but it can't recover from all failures. For instance, if the client doesn't send anything for a while, and when the server starts to send the locator pair doesn't work any more. In this case the server can do nothing since it doesn't have a context with alternate locators, and the client can't possibly know that the server might be having problems reaching it.
- o When does a host need to verify the locator list? Immediately i.e. before accepting packets from those locators as the source address? Or before sending packets to those locators? There are some issues if it isn't verified immediately since it allows an on-path attacker to send bogus update messages which can not be verified; that would potentially make the host no longer accept packets from the actual locator that the peer is using, and when it tries to verify the locators it would find that they are "bad" and has no alternate peer locator it can use. This is the case even if the peer has sent a locator list as long as the attacker has sent a more recent one.

18. Implications Elsewhere

The general shim6 approach, as well as the specifics of this proposed solution, has implications elsewhere. The key implications are:

- o Applications that perform referrals, or callbacks using IP addresses as the 'identifiers' can still function in limited ways, as described in [18]. But in order for such applications to be able to take advantage of the multiple locators for redundancy, the applications need to be modified to either use fully qualified domain names as the 'identifiers', or they need to pass all the

locators as the 'identifiers' i.e., the 'identifier' from the applications perspective becomes a set of IP addresses instead of a single IP address.

- Firewalls that today pass limited traffic, e.g., outbound TCP connections, would presumably block the shim6 protocol. This means that even when shim6 capable hosts are communicating, the I1 messages would be dropped, hence the hosts would not discover that their peer is shim6 capable. This is in fact a feature, since if the hosts managed to establish a host-pair context, then the firewall would probably drop the "different" packets that are sent after a failure (those using the shim6 payload message with a TCP packet inside it). Thus stateful firewalls that are modified to allow shim6 messages through should also be modified to allow the payload messages through after a failure. This presumably implies that the firewall needs to track the set of locators in use by looking at the shim6 exchanges. Such firewalls might even want to verify the locators using the HBA/CGA verification themselves.
- Signaling protocols for QoS or other things that involve having devices in the network path look at IP addresses and port numbers, or IP addresses and Flow Labels, need to be invoked on the hosts when the locator pair changes due to a failure. At that point in time those protocols need to inform the devices that a new pair of IP addresses will be used for the flow. Note that this is the case even though we no longer overload the flow label as a context tag; the in-path devices need to know about the use of the new locators even though the flow label stays the same.
- MTU implications. The path MTU mechanisms we use are robust against different packets taking different paths through the Internet, by computing a minimum over the recently observed path MTUs. When shim6 fails over from using one locator pair to another pair, this means that packets might travel over a different path through the Internet, hence the path MTU might be quite different. Perhaps such a path change would be a good hint to the path MTU mechanism to try a larger MTU?

  The fact that the shim, at least for uncommon payload types, will add an 8 octet extension header (the payload message) after a locator switch, can also affect the usable path MTU for the ULPs. In this case the MTU change is local to the sending host, thus conveying the change to the ULPs is an implementation matter.

19. Security Considerations

This document satisfies the concerns specified in [17] as follows:

- TBD: Using HBA or CGA for ...

Some of the residual threats in this proposal are:

- An attacker which arrives late on the path (after the context has been established) can use the No Such Context error to cause one peer to recreate the context, and at that point in time the attacker can observe all of the exchange. But this doesn't seem to open any new doors for the attacker since such an attacker can observe the Context tags that are being used, and once known it can use those to send bogus messages.
- An attacker which is present on the path so that it can find out the context tags, can generate a No Such Context error after it has moved off the path. For this packet to be effective it needs to have a source locator which belongs to the context, thus there can not be "too much" ingress filtering between the attackers new location and the communicating peers. But this doesn't seem to be that severe, because once the error causes the context to be torn down and re-established, a new pair of context tags will be used, which will not be known to the attacker. If this is still a concern, we could require a 2-way handshake "did you really loose the state?" in response to the error message.
- It might be possible for an attacker to try random 32-bit context tags and see if they can cause disruption for communication between two hosts. We can make this harder by using a larger context tag; 47 bits is the largest that fit in the 8-octet payload header. If this isn't sufficient, one could use an even larger tag in the shim6 control messages, and use the low-order 47 bits in the payload header.

20. IANA Considerations

IANA needs to allocate a new IP Next Header value for this protocol.

IANA also needs to record a CGA message type for this protocol in the [CGA] namespace, 0x4A30 5662 4858 574B 3655 416F 506A 6D48.

TBD: the IANA rules for the shim6 message types and option types.

21. Possible Protocol Extensions

During the development of this protocol, several issues have been brought up as important one to address, but are ones that do not need to be in the base protocol itself but can instead be done as extensions to the protocol. The key ones are:

- Is there need for keeping the list of locators private between the two communicating endpoints? We can potentially accomplish that when using CGA but not with HBA, but it comes at the cost of doing some public key encryption and decryption operations as part of the context establishment. The suggestion is to leave this for a future extension to the protocol.

o Defining some form of end-to-end "compression" mechanism that removes the need for including the Shim6 Payload extension header when the locator pair is not the ULID pair.

22. Change Log

The following changes have been made since draft-ietf-shim6-proto-00:

o Removed the use of the flow label and the overloading of the IP protocol numbers. Instead, when the locator pair is not the ULID pair, the ULP payloads will be carried with an 8 octet extension header. The belief is that it is possible to remove these extra bytes by defining future shim6 extensions that exchange more information between the hosts, without having to overload the flow label or the IP protocol numbers.

o Grew the context tag from 20 bits to 32 bits, with the possibility to grow it to 47 bits. This implies changes to the message formats.

o Almost by accident, the new shim6 message format is very close to the HIP message format.

o Adopted the HIP format for the options, since this makes it easier to describe variable length options. The original, ND-style, option format requires internal padding in the options to make them 8 octet length in total, while the HIP format handles that using the option length field.

o Removed some of the control messages, and renamed the other ones.

o Added a "generation" number to the Locator List option, so that the peers can ensure that the preferences refer to the right "version" of the Locator List.

o In order for FBD and exploration to work when there the use of the context is forked, that is different ULP messages are sent over different locator pairs, things are a lot easier if there is only one current locator pair used for each context. Thus the forking of the context is now causing a new context to be established for the same ULID; the new context having a new context tag. The original context is referred to as the "default" context for the ULID pair.

o Added more background material and textual descriptions.

23. Acknowledgements

Over the years many people active in the multi6 and shim6 WGs have contributed ideas a suggestions that are reflected in this draft.

Thanks to Marcelo Bagnulo for providing comments on earlier versions of this draft.

Appendix A. Design Alternatives

This document has picked a certain set of design choices in order to try to work out a bunch of the details, and stimulate discussion. But as has been discussed on the mailing list, there are other choices that make sense. This appendix tries to enumerate some alternatives.

Appendix A.1 Context granularity

TBD

Appendix A.2 Demultiplexing of data packets in shim6 communications

Once a Host-pair context is established between two hosts, packets may carry locators that differ from the ULIDs presented to the ULPs using the established context. One of main functions of the SHIM6 layer is to perform the mapping between the locators used to forward packets through the network and the ULIDs presented to the ULP. In order to perform that translation for incoming packets, the SHIM6 layer needs to first identify which of the incoming packets need to be translated and then perform the mapping between locators and ULIDs using the associated context. Such operation is called demultiplexing. It should be noted that because any address can be used both as a locator and as a ULID, additional information other than the addresses carried in packets, need to be taken into account for this operation.

For example, if a host has address A1 and A2 and starts communicating with a peer with addresses B1 and B2, then some communication (connections) might use the pair <A1, B1> as ULID and others might use e.g., <A2, B2>. Initially there are no failures so these address pairs are used as locators i.e. in the IP address fields in the packets on the wire. But when there is a failure the shim6 layer on A might decide to send packets that used <A1, B1> as ULIDs using <A2, B2> as the locators. In this case B needs to be able to rewrite the IP address field for some packets and not others, but the packets all have the same locator pair.

In order to accomplish the demultiplexing operation successfully, data packets carry a context tag that allows the receiver of the packet to determine the shim context to be used to perform the operation.

Two mechanisms for carrying the context tag information have been considered in depth during the shim protocol design. Those carrying the context tag in the flow label field of the IPv6 header and the usage of a new extension header to carry the context tag. In this

appendix we will describe the pros and cons of each approach and justify the selected option.

Appendix A.2.1 Flow-label

A possible approach is to carry the context tag in the Flow Label field of the IPv6 header. This means that when a shim6 context is established, a Flow Label value is associated with this context (and perhaps a separate flow label for each direction).

The simplest approach that does this is to have the triple <Flow Label, Source Locator, Destination Locator> identify the context at the receiver.

The problem with this approach is that because the locator sets are dynamic, it is not possible at any given moment to be sure that two contexts for which the same context tag is allocated will have disjoint locator sets during the lifetime of the contexts.

Suppose that Node A has addresses IPA1, IPA2, IPA3 and IPA4 and that Host B has addresses IPB1 and IPB2.

Suppose that two different contexts are established between HostA and HostB.

Context #1 is using IPA1 and IPB1 as ULIDs. The locator set associated to IPA1 is IPA1 and IPA2 while the locator set associated to IPB1 is just IPB1.

Context #2 uses IPA3 and IPB2 as ULIDs. The locator set associated to IPA3 is IPA3 and IPA4 and the locator set associated to IPB2 is just IPB2.

Because the locator sets of the Context #1 and Context # 2 are disjoint, hosts could think that the same context tag value can be assigned to both of them. The problem arrives when later on IPA3 is added as a valid locator for IPA1 and IPB2 is added as a valid locator for IPB1 in Context #1. In this case, the triple <Flow Label, Source Locator, Destination Locator> would not identify a unique context anymore and correct demultiplexing is no longer possible.

A possible approach to overcome this limitation is simply not to repeat the Flow Label values for any communication established in a host. This basically means that each time a new communication that is using different ULIDs is established, a new Flow Label value is assigned to it. By this mean, each communication that is using different ULIDs can be differentiated because it has a different Flow

Label value.

The problem with such approach is that it requires that the receiver of the communication allocates the Flow Label value used for incoming packets, in order to assign them uniquely. For this, a shim negotiation of the Flow Label value to use in the communication is needed before exchanging data packets. This poses problems with non-shim capable hosts, since they would not be able to negotiate an acceptable value for the Flow Label. This limitation can be lifted by marking the packets that belong to shim sessions from those that do not. These marking would require at least a bit in the IPv6 header that is not currently available, so more creative options would be required, for instance using new Next Header values to indicate that the packet belongs to a shim6 enabled communication and that the Flow Label carries context information as proposed in the now expire NOID draft. . However, even if this is done, this approach is incompatible with the deferred establishment capability of the shim protocol, which is a preferred function, since it suppresses the delay due to the shim context establishment prior to initiation of the communication and it also allows nodes to define at which stage of the communication they decide, based on their own policies, that a given communication requires to be protected by the shim.

In order to cope with the identified limitations, an alternative approach that does not constraints the flow label values used by communications that are using ULIDs equal to the locators (i.e. no shim translation) is to only require that different flow label values are assigned to different shim contexts. In such approach communications start with unmodified flow label usage (could be zero, or as suggested in [15]). The packets sent after a failure when a different locator pair is used would use a completely different flow label, and this flow label could be allocated by the receiver as part of the shim context establishment. Since it is allocated during the context establishment, the receiver of the "failed over" packets can pick a flow label of its choosing (that is unique in the sense that no other context is using it as a context tag), without any performance impact, and respecting that for each locator pair, the flow label value used for a given locator pair doesn't change due to the operation of the multihoming shim.

In this approach, the constraint is that Flow Label values being used as context identifiers cannot be used by other communications that use non-disjoint locator sets. This means that once that a given Flow Label value has been assigned to a shim context that has a certain locator sets associated, the same value cannot be used for other communications that use an address pair that is contained in the locator sets of the context. This is a constraint in the

potential Flow Label allocation strategies.

A possible workaround to this constraint is to mark shim packets that require translation, in order to differentiate them from regular IPv6 packets, using the artificial Next Header values described above. In this case, the Flow Label values constrained are only those of the packets that are being translated by the shim. This last approach would be the preferred approach if the context tag is to be carried in the Flow Label field. This is not only because it imposes the minimum constraints to the Flow Label allocation strategies, limiting the restrictions only to those packets that need to be translated by the shim, but also because Context Loss detection mechanisms greatly benefit from the fact that shim data packets are identified as such, allowing the receiving end to identify if a shim context associated to a received packet is suppose to exist, as it will be discussed in the Context Loss detection appendix below.

Appendix A.2.2 Extension Header

Another approach is to carry the context tag in a new Extension Header. These context tags are allocated by the receiving end during the shim6 protocol initial negotiation, implying that each context will have two context tags, one for each direction. Data packets will be demultiplexed using the context tag carried in the Extension Header. This seems a clean approach since it does not overload existing fields. However, it introduces additional overhead in the packet due to the additional header. The additional overhead introduced is 8 octets. However, it should be noted that the context tag is only required when a locator other than the one used as ULID is contained in the packet. Packets where both the source and destination address fields contain the ULIDs do not require a context tag, since no rewriting is necessary at the receiver. This approach would reduce the overhead, because the additional header is only required after a failure. On the other hand, this approach would cause changes in the available MTU for some packets, since packets that include the Extension Header will have an MTU 8 octets shorter.

Appendix A.3 Context Loss Detection

In this appendix we will present different approaches considered to detect context loss and potential context recovery strategies. The scenario being considered is the following: Node A and Node B are communicating using IPA1 and IPB1. Sometime later, a shim context is established between them, with IPA1 and IPB1 as ULIDs and IPA1,...,IPAn and IPB1,...,IPBm as locator set respectively.

It may happen, that later on, one of the hosts, e.g. Host A looses the shim context. The reason for this can be that Host A has a more

aggressive garbage collection policy than HostB or that an error occurred in the shim layer at host A resulting in the loss of the context state.

The mechanisms considered in this appendix are aimed to deal with this problem. There are essentially two tasks that need to be performed in order to cope with this problem: first, the context loss must be detected and second the context needs to be recovered/ reestablished.

Mechanisms for detecting context. loss

These mechanisms basically consist in that each end of the context periodically sends a packet containing context-specific information to the other end. Upon reception of such packets, the receiver verifies that the required context exists. In case that the context does not exist, it sends a packet notifying the problem to the sender.

An obvious alternative for this would be to create a specific context keepalive exchange, which consists in periodically sending packets with this purpose. This option was considered and discarded because it seemed an overkill to define a new packet exchange to deal with this issue.

An alternative is to piggyback the context loss detection function in other existent packet exchanges. In particular, both shim control and data packets can be used for this.

Shim control packets can be trivially used for this, because they carry context specific information, so that when a node receives one of such packets, it will verify if the context exists. However, shim control frequency may not be adequate for context loss detection since control packet exchanges can be very limited for a session in certain scenarios.

Data packets, on the other hand, are expected to be exchanged with a higher frequency but they do not necessarily carry context specific information. In particular, packets flowing before a locator change (i.e. packet carrying the ULIDs in the address fields) do not need context information since they do not need any shim processing. Packets that carry locators that differ from the ULIDs carry context information.

However, we need to make a distinction here between the different approaches considered to carry the context tag, in particular between those approaches where packets are explicitly marked as shim packets and those approaches where packets are not marked as such. For

instance, in the case where the context tag is carried in the Flow Label and packets are not marked as shim packets (i.e. no new Next Header values are defined for shim), a receiver that has lost the associated context is not able to detect that the packet is associated with a missing context. The result is that the packet will be passed unchanged to the upper layer protocol, which in turn will probably silently discard it due to a checksum error. The resulting behavior is that the context loss is undetected. This is one additional reason to discard an approach that carries the context tag in the Flow Label field and does not explicitly mark the shim packets as such. On the other hand, approaches that mark shim data packets (like the Extension Header or the Flow Label with new Next Header values approaches) allow the receiver to detect if the context associated to the received packet is missing. In this case, data packets also perform the function of a context loss detection exchange. However, it must be noted that only those packets that carry a locator that differs form the ULID are marked. This basically means that context loss will be detected after an outage has occurred i.e. alternative locators are being used.

Summarizing, the proposed context loss detection mechanisms uses shim control packets and payload packets to detect context loss. Shim control packets detect context loss during the whole lifetime of the context, but the expected frequency in some cases is very low. On the other hand, payload packets have a higher expected frequency in general, but they only detect context loss after an outage. This behavior implies that it will be common that context loss is detected after a failure i.e. once that it is actually needed. Because of that, a mechanism for recovering from context loss is required if this approach is used.

Overall, the mechanism for detecting lost context would work as follows: the end that still has the context available sends a message referring to the context. Upon the reception of such message, the end that has lost the context identifies the situation and notifies the context loss event to the other end by sending a packet containing the lost context information extracted from the received packet.

One option is to simply send an error message containing the received packets (or at least as much of the received packet that the MTU allows to fit in). One of the goals of this notification is to allow the other end that still retains context state, to reestablish the lost context. The mechanism to reestablish the loss context consists in performing the 4-way initial handshake. This is a time consuming exchange and at this point time may be critical since we are reestablishing a context that is currently needed (because context loss detection may occur after a failure). So, another option is to

replace the error message by a modified R1 message, so that the time required to perform the context establishment exchange can be reduced. Upon the reception of this modified R1 message, the end that still has the context state can finish the context establishment exchange and restore the lost context.

Appendix A.4 Securing locator sets

The adoption of a protocol like SHIM that allows the binding of a given ULID with a set of locators opens the doors for different types of redirection attacks as described in [17]. The goal in terms of security for the design of the shim protocol is not to introduce any new vulnerability in the Internet architecture. It is a non-goal to provide additional protection than the currently available in the single-homed IPv6 Internet.

Multiple security mechanisms were considered to protect the shim protocol. In this appendix we will present some of them.

The simplest option to protect the shim protocol was to use cookies i.e. a randomly generated bit string that is negotiated during the context establishment phase and then it is included in following signaling messages. By this mean, it would be possible to verify that the party that was involved in the initial handshake is the same party that is introducing new locators. Moreover, before using a new locator, an exchange is performed through the new locator, verifying that the party located at the new locator knows the cookie i.e. that it is the same party that performed the initial handshake.

While this security mechanisms does indeed provide a fair amount of protection, it does leave the door open for the so-called time shifted attacks. In these attacks, an attacker that once was on the path, it discovers the cookie by sniffing any signaling message. After that, the attacker can leave the path and still perform a redirection attack, since as he is in possession of the cookie, he can introduce a new locator in the locator set and he can also successfully perform the reachability exchange if he is able to receive packets at the new locator. The difference with the current single-homed IPv6 situation is that in the current situation the attacker needs to be on-path during the whole lifetime of the attack, while in this new situation where only cookie protection if provided, an attacker that once was on the path can perform attacks after he has left the on-path location.

Moreover, because the cookie is included in signaling messages, the attacker can discover the cookie by sniffing any of them, making the protocol vulnerable during the whole lifetime of the shim context. A possible approach to increase the security was to use a shared secret

i.e. a bit string that is negotiated during the initial handshake but that is used as a key to protect following messages. With this technique, the attacker must be present on the path sniffing packets during the initial handshake, since it is the only moment where the shared secret is exchanged. While this improves the security, it is still vulnerable to time shifted attacks, even though it imposes that the attacker must be on path at a very specific moment (the establishment phase) to actually be able to launch the attack. While this seems to substantially improve the situation, it should be noted that, depending on protocol details, an attacker may be able to force the recreation of the initial handshake (for instance by blocking messages and making the parties think that the context has been lost), so the resulting situation may not differ that much from the cookie based approach.

Another option that was discussed during the design of the protocol was the possibility of using IPSec for protecting the shim protocol. Now, the problem under consideration in this scenario is how to securely bind an address that is being used as ULID with a locator set that can be used to exchange packets. The mechanism provided by IPSec to securely bind the address used with the cryptographic keys is the usage of digital certificates. This implies that an IPSec based solution would require that the generation of digital certificates that bind the key and the ULID by a common third trusted party for both parties involved in the communication. Considering that the scope of application of the shim protocol is global, this would imply a global public key infrastructure. The major issues with this approach are the deployment difficulties associated with a global PKI.

Finally two different technologies were selected to protect the shim protocol: HBA [6] and CGA [5]. These two approaches provide a similar level of protection but they provide different functionality with a different computational cost.

The HBA mechanism relies on the capability of generating all the addresses of a multihomed host as an unalterable set of intrinsically bound IPv6 addresses, known as an HBA set. In this approach, addresses incorporate a cryptographic one-way hash of the prefix-set available into the interface identifier part. The result is that the binding between all the available addresses is encoded within the addresses themselves, providing hijacking protection. Any peer using the shim protocol node can efficiently verify that the alternative addresses proposed for continuing the communication are bound to the initial address through a simple hash calculation. A limitation of the HBA technique is that once generated the address set is fixed and cannot be changed without also changing all the addresses of the HBA set. In other words, the HBA technique does not support dynamic

addition of address to a previously generated HBA set. An advantage of this approach is that it requires only hash operations to verify a locator set, imposing very low computational cost to the protocol.

In a CGA based approach the address used as ULID is a CGA that contains a hash of a public key in its interface identifier. The result is a secure binding between the ULID and the associated key pair. This allows each peer to use the corresponding private key to sign the shim messages that convey locator set information. The trust chain in this case is the following: the ULID used for the communication is securely bound to the key pair because it contains the hash of the public key, and the locator set is bound to the public key through the signature. The CGA approach then supports dynamic addition of new locators in the locator set, since in order to do that, the node only needs to sign the new locator with the private key associated with the CGA used as ULID. A limitation of this approach is that it imposes systematic usage of public key cryptography with its associate computational cost.

Any of these two mechanisms HBA and CGA provide time-shifted attack protection, since the ULID is securely bound to a locator set that can only be defined by the owner of the ULID.

So, the design decision adopted was that both mechanisms HBA and CGA are supported, so that when only stable address sets are required, the nodes can benefit from the low computational cost offered by HBA while when dynamic locator sets are required, this can be achieved through CGAs with an additional cost. Moreover, because HBAs are defined as a CGA extension, the addresses available in a node can simultaneously be CGAs and HBAs, allowing the usage of the HBA and CGA functionality when needed without requiring a change in the addresses used.

Appendix A.5 Host-pair context establishment exchange

Two options were considered for the host-pair context establishment exchange: a 2-way handshake and a 4-way handshake.

A key goal for the design of this exchange was that protection against DoS attacks. The attack under consideration was basically a situation where an attacker launches a great amount of host-pair establishment request packets, exhausting victim's resources, similar to TCP SYN flooding attacks.

A 4 way-handshake exchange protects against these attacks because the receiver does not creates any state associate to a given context until the reception of the second packet which contains a prior contact proof in the form of a token. At this point the receiver can

verify that at least the address used by the initiator is at some extent valid, since the initiator is able to receive packets at this address. In the worse case, the responder can track down the attacker using this address. The drawback of this approach is that it imposes a 4 packet exchange for any context establishment. This would be a great deal if the shim context needed to be established up front, before the communication can proceed. However, thanks to deferred context establishment capability of the shim protocol, this limitation has a reduced impact in the performance of the protocol. (It may however have a greater impact in the situation of context recover as discussed earlier, but in this case, it is possible to perform optimizations to reduce the number of packets as described above)

The other option considered was a 2-way handshake with the possibility to fall back to a 4-way handshake in case of attack. In this approach, the host pair establishment exchange normally consists in a 2-packet exchange and it does not verify that the initiator has performed a prior contact before creating context state. In case that a DoS attack is detected, the responder falls back to a 4-way handshake similar to the one described previously in order to prevent the detected attack to proceed. The main difficulty with this attack is how to detect that a responder is currently under attack. It should be noted, that because this is 2-way exchange, it is not possible to use the number of half open sessions (as in TCP) to detect an ongoing attack and different heuristics need to be considered.

The design decision taken was that considering the current impact of DoS attacks and the low impact of the 4-way exchange in the shim protocol thanks to the deferred context establishment capability, a 4-way exchange would be adopted for the base protocol.

Appendix A.6 **Updating locator sets**

There are two possible approaches to the addition and removal of locators: atomic and differential approaches. The atomic approach essentially send the complete locators set each time that a variation in the locator set occurs. The differential approach send the differences between the existing locator set and the new one. The atomic approach imposes additional overhead, since all the locator set has to be exchanged each time while the differential approach requires re-synchronization of both ends through changes i.e. that both ends have the same idea about what the current locator set is.

Because of the difficulties imposed by the synchronization requirement, the atomic approach was selected.

Appendix A.7 State Cleanup

There are essentially two approaches for discarding an existing state about locators, keys and identifiers of a correspondent node: a coordinated approach and an unilateral approach.

In the unilateral approach, each node discards the information about the other node without coordination with the other node based on some local timers and heuristics. No packet exchange is required for this. In this case, it would be possible that one of the nodes has discarded the state while the other node still hasn't. In this case, a No-Context error message may be required to inform about the situation and possibly a recovery mechanism is also needed.

A coordinated approach would use an explicit CLOSE mechanism, akin to the one specified in HIP [23]. If an explicit CLOSE handshake and associated timer is used, then there would no longer be a need for the No Context Error message due to a peer having garbage collected its end of the context. However, there is still potentially a need to have a No Context Error message in the case of a complete state loss of the peer (also known as a crash followed by a reboot). Only if we assume that the reboot takes at least the CLOSE timer, or that it is ok to not provide complete service until CLOSE timer minutes after the crash, can we completely do away with the No Context Error message.

24. References

24.1 Normative References

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

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

[3] Narten, T., Nordmark, E., and W. Simpson, "Neighbor Discovery for IP Version 6 (IPv6)", RFC 2461, December 1998.

[4] Thomson, S. and T. Narten, "IPv6 Stateless Address Autoconfiguration", RFC 2462, December 1998.

[5] Aura, T., "Cryptographically Generated Addresses (CGA)", RFC 3972, March 2005.

[6] Bagnulo, M., "Hash Based Addresses (HBA)", draft-ietf-shim6-hba-00 (work in progress), July 2005.

[7] Beijnum, I., "Shim6 Reachability Detection", draft-ietf-shim6-reach-detect-00 (work in progress), July 2005.

[8] Arkko, J., "Failure Detection and Locator Pair Exploration Design for IPv6 Multihoming", draft-ietf-shim6-failure-detection-01 (work in progress), October 2005.

24.2 Informative References

[9] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, February 2000.

[10] Ferguson, P. and D. Senie, "Network Ingress Filtering: Defeating Denial of Service Attacks which employ IP Source Address Spoofing", BCP 38, RFC 2827, May 2000.

[11] Narten, T. and R. Draves, "Privacy Extensions for Stateless Address Autoconfiguration in IPv6", RFC 3041, January 2001.

[12] Draves, R., "Default Address Selection for Internet Protocol version 6 (IPv6)", RFC 3484, February 2003.

[13] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

[14] Abley, J., Black, B., and V. Gill, "Goals for IPv6 Site-Multihoming Architectures", RFC 3582, August 2003.

[15] Rajahalme, J., Conta, A., Carpenter, B., and S. Deering, "IPv6 Flow Label Specification", RFC 3697, March 2004.

[16] Hinden, R. and B. Haberman, "Unique Local IPv6 Unicast Addresses", RFC 4193, October 2005.

[17] Nordmark, E., "Threats relating to IPv6 multihoming solutions", draft-ietf-multi6-multihoming-threats-03 (work in progress), January 2005.

[18] Nordmark, E., "Shim6 Application Referral Issues", draft-ietf-shim6-app-refer-00 (work in progress), July 2005.

[19] Abley, J., "Shim6 Applicability Statement", draft-ietf-shim6-applicability-00 (work in progress), July 2005.

[20] Huston, G., "Architectural Commentary on Site Multi-homing using a Level 3 Shim", draft-ietf-shim6-arch-00 (work in progress), July 2005.

[21] Bagnulo, M. and J. Arkko, "Functional decomposition of the multihoming protocol", draft-ietf-shim6-functional-dec-00 (work in progress), July 2005.

[22] Nordmark, E. and M. Bagnulo, "Multihoming L3 Shim Approach", draft-ietf-shim6-l3shim-00 (work in progress), July 2005.

[23] Moskowitz, R., "Host Identity Protocol", draft-ietf-hip-base-03 (work in progress), June 2005.

[24] Lear, E. and R. Droms, "What's In A Name:Thoughts from the NSRG", draft-irtf-nsrg-report-10 (work in progress), September 2003.

Author's Address

Erik Nordmark
Sun Microsystems
17 Network Circle
Menlo Park, CA 94025
USA

Phone: +1 650 786 2921
Email: erik.nordmark@sun.com

Acknowledgment

Funding for the RFC Editor function is currently provided by the Internet Society.