Internet Engineering Task Force (IETF)
Request for Comments: 8003
Obsoletes: 5203
Category: Standards Track
ISSN: 2070-1721
October 2016

J. Laganier
Luminate Wireless, Inc.
L. Eggert
NetApp

Host Identity Protocol (HIP) Registration Extension

Abstract

This document specifies a registration mechanism for the Host Identity Protocol (HIP) that allows hosts to register with services, such as HIP rendezvous servers or middleboxes. This document obsoletes RFC 5203.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

This document specifies an extension to the Host Identity Protocol (HIP) [RFC7401]. The extension provides a generic means for a host to register with a service. The service may, for example, be a HIP rendezvous server [RFC8004] or a middlebox [RFC3234].

This document makes no further assumptions about the exact type of service. Likewise, this document does not specify any mechanisms to discover the presence of specific services or means to interact with them after registration. Future documents may describe those operations.

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in RFC 2119 [RFC2119].

2. Terminology

In addition to the terminology defined in the HIP Architecture [HIP-ARCH], the HIP specification [RFC7401], and the HIP Rendezvous Extension [RFC8004], this document defines and uses the following terms:

Requester:
a HIP node registering with a HIP registrar to request registration for a service.

Registrar:
a HIP node offering registration for one or more services.

Service:
a facility that provides requesters with new capabilities or functionalities operating at the HIP layer. Examples include firewalls that support HIP traversal or HIP rendezvous servers.

Registration:
shared state stored by a requester and a registrar, allowing the requester to benefit from one or more HIP services offered by the registrar. Each registration has an associated finite lifetime. Requesters can extend established registrations through re-registration (i.e., perform a refresh).

Registration Type:
an 8-bit identifier for a given service in the registration protocol. For example, the rendezvous service is identified by a specific registration type.

3. HIP Registration Extension Overview

This document does not specify the means by which a requester discovers the availability of a service or how a requester locates a registrar. After a requester has discovered a registrar, it either initiates HIP base exchange or uses an existing HIP association with the registrar. In both cases, registrars use additional parameters, which the remainder of this document defines, to announce their quality and grant or refuse registration. Requesters use corresponding parameters to register with the service. Both the registrar and the requester MAY also include in the messages exchanged additional HIP parameters specific to the registration type requested. Other documents will define parameters and how they shall be used.

The HIP base exchange, including the definition of the HIP I1, R1, I2, and R2 packets, is defined in [RFC7401]. The following sections describe the differences between this registration handshake and the standard HIP base exchange [RFC7401].

3.1. Registrar Announcing Its Ability

A host that is capable and willing to act as a registrar vis-a-vis a specific requester SHOULD include a REG_INFO parameter in the R1 packets it sends during all base exchanges with that requester. If it is currently unable to provide services due to transient conditions, it SHOULD include an empty REG_INFO, i.e., one with no services listed. If services can be provided later, it SHOULD send UPDATE packets indicating the current set of services available in a new REG_INFO parameter to all hosts it is associated with.

3.2. Requester Requesting Registration

To request registration with a service, a requester constructs and includes a corresponding REG_REQUEST parameter in an I2 or UPDATE packet it sends to the registrar.

If the requester has no HIP association established with the registrar, it SHOULD send the REG_REQUEST at the earliest possibility, i.e., in the I2 packet. This minimizes the number of packets that need to be exchanged with the registrar. A registrar MAY end a HIP association that does not carry a REG_REQUEST by including a NOTIFY with the type REG_REQUIRED in the R2. In this case, no HIP association is created between the hosts. The REG_REQUIRED notification error type is 51.

3.3. Registrar Granting or Refusing Service(s) Registration

Once registration has been requested, the registrar is able to authenticate the requester based on the host identity included in I2.

If the registrar knows the Host Identities (HIs) of all the hosts that are allowed to register for service(s), it SHOULD reject registrations from unknown hosts. However, since it may be infeasible to preconfigure the registrar with all the HIs, the registrar SHOULD also support HIP certificates [RFC8002] to allow for certificate-based authentication.

When a requester wants to register with a registrar, it SHOULD check if it has a suitable certificate for authenticating with the registrar. How the suitability is determined and how the certificates are obtained is out of scope for this document. If the requester has one or more suitable certificates, the host SHOULD include them (or just the most suitable one) in a CERT parameter to the HIP packet along with the REG_REQUEST parameter. If the requester does not have any suitable certificates, it SHOULD send the registration request without the CERT parameter to test whether the registrar accepts the request based on the host's identity.

When a registrar receives a HIP packet with a REG_REQUEST parameter, and it requires authentication for at least one of the registration types listed in the REG_REQUEST parameter, it MUST first check whether the HI of the requester is in the allowed list for all the registration types in the REG_REQUEST parameter. If the requester is in the allowed list (or the registrar does not require any authentication), the registrar MUST proceed with the registration.

If the requester was not in the allowed list and the registrar requires the requester to authenticate, the registrar MUST check whether the packet also contains a CERT parameter. If the packet does not contain a CERT parameter, the registrar MUST reject the registrations requiring authentication with Failure Type 0 (zero) (registration requires additional credentials). If the certificate is valid and accepted (issued for the requester and signed by a trusted issuer), the registrar MUST proceed with the registration. If the certificate in the parameter is not accepted, the registrar MUST reject the corresponding registrations with the appropriate Failure Type:

4 (Bad certificate): The certificate is corrupt, contains invalid signatures, etc.

5 (Unsupported certificate): The certificate is of an unsupported type.

6 (Certificate expired): The certificate is no longer valid.

7 (Certificate other): The certificate could not be validated for some unspecified reason.

8 (Unknown CA): The issuing certification authority (CA) certificate could not be located or is not trusted.

After successful authorization, the registrar includes a REG_RESPONSE parameter in its response, which contains the service type(s) for which it has authorized registration, and zero or more REG_FAILED parameters containing the service type(s) for which it has not authorized registration or registration has failed for other reasons. This response can be either an R2 or an UPDATE message, respectively, depending on whether the registration was requested during the base exchange or using an existing association. In particular, REG_FAILED with a Failure Type of zero indicates the service type(s) that requires further credentials for registration.

If the registrar requires further authorization and the requester has additional credentials available, the requester SHOULD try to register again with the service after the HIP association has been established.

Successful processing of a REG_RESPONSE parameter creates registration state at the requester. In a similar manner, successful processing of a REG_REQUEST parameter creates registration state at the registrar and possibly at the service. Both the requester and registrar can cancel a registration before it expires, if the services afforded by a registration are no longer needed by the requester or cannot be provided any longer by the registrar (for instance, because its configuration has changed).

```
             +-----+          I1          +-----+-----+
             |     |--------------------->|     |  S1 |
             |     |<---------------------|     |     |
             |     | R1(REG_INFO:S1,S2,S3)|     +-----+
             | RQ  |                      |  R  |  S2 |
             |     |    I2(REG_REQ:S1)    |     |     |
             |     |--------------------->|     +-----+
             |     |<---------------------|     |  S3 |
             |     |    R2(REG_RESP:S1)   |     |     |
             +-----+                      +-----+-----+
```

A requester (RQ) registers for service (S1) with a registrar (R) of services (S1), (S2), and (S3) with which it has no current HIP association

```
             +-----+                      +-----+-----+
             |     |  UPDATE(REG_INFO:S)  |     |     |
             |     |<---------------------|     |     |
             | RQ  |--------------------->|  R  |  S  |
             |     |  UPDATE(REG_REQ:S)   |     |     |
             |     |  UPDATE(REG_RESP:S)  |     |     |
             |     |<---------------------|     |     |
             +-----+                      +-----+-----+
```

A requester (RQ) registers for service (S) with a registrar (R) of services (S) with which it currently has a HIP association established

4. Parameter Formats and Processing

This section describes the format and processing of the new parameters introduced by the HIP Registration Extension. The encoding of these new parameters conforms to the HIPv2 TLV format described in Section 5.2.1 of RFC7401 [RFC7401].

4.1. Encoding Registration Lifetimes with Exponents

The HIP registration uses an exponential encoding of registration lifetimes.

The special value 0 (zero) of the lifetime field MUST be interpreted as representing a special lifetime duration of 0 (zero) seconds and is used to request and grant cancellation of a registration.

The non-zero values of the lifetime field used throughout this document MUST be interpreted as an exponent value representing a lifetime duration of 2^((lifetime - 64)/8) seconds.

This allows a compact encoding of 255 different lifetime durations (in addition to the special lifetime duration of zero seconds) ranging from 2^(63/8) seconds (i.e., ~4 ms) to 2^(191/8) seconds (i.e., ~178 days) into an 8-bit integer field.

4.2. REG_INFO

```
 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              |             Length            |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
| Min Lifetime  | Max Lifetime  |  Reg Type #1  |  Reg Type #2  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |     ...       |  Reg Type #n  |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    Padding    +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type	930
Length	Length in octets, excluding Type, Length, and Padding.
Min Lifetime	Minimum registration lifetime.
Max Lifetime	Maximum registration lifetime.
Reg Type	The registration types offered by the registrar.

Other documents will define specific values for registration types. See Section 7 for more information.

Registrars include the parameter in R1 packets in order to announce their registration capabilities. The registrar SHOULD include the parameter in UPDATE packets when its service offering has changed. HIP_SIGNATURE_2 protects the parameter within the R1 packets.

The registrar indicates the minimum and maximum registration lifetime that it is willing to offer to a requester. A requester SHOULD NOT request registration with a lifetime greater than the maximum registration lifetime or smaller than the minimum registration lifetime.

4.3. REG_REQUEST

```
 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              |           Length              |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Lifetime    |  Reg Type #1  |  Reg Type #2  |  Reg Type #3  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |     ...       |  Reg Type #n  |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    Padding    +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type	932
Length	Length in octets, excluding Type, Length, and Padding.
Lifetime	Requested registration lifetime.
Reg Type	The preferred registration types in order of preference.

Other documents will define specific values for registration types. See Section 7 for more information.

A requester includes the REG_REQUEST parameter in I2 or UPDATE packets to register with a registrar's service(s). If the REG_REQUEST parameter is in an UPDATE packet, the registrar MUST NOT modify the registrations of registration types that are not listed in the parameter. Moreover, the requester MUST NOT include the parameter unless the registrar's R1 packet or latest received UPDATE packet has contained a REG_INFO parameter with the requested registration types.

The requester MUST NOT include more than one REG_REQUEST parameter in its I2 or UPDATE packets, while the registrar MUST be able to process one or more REG_REQUEST parameters in received I2 or UPDATE packets.

When the registrar receives a registration with a lifetime that is either smaller or greater than the minimum or maximum lifetime, respectively, then it SHOULD grant the registration for the minimum or maximum lifetime, respectively.

HIP_SIGNATURE protects the parameter within the I2 and UPDATE packets.

4.4. REG_RESPONSE

```
 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              |            Length             |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|   Lifetime    |  Reg Type #1  |  Reg Type #2  |  Reg Type #3  |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
|      ...      |      ...      |  Reg Type #n  |               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    Padding    +
|                                                               |
+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type	934
Length	Length in octets, excluding Type, Length, and Padding.
Lifetime	Granted registration lifetime.
Reg Type	The granted registration types in order of preference.

Other documents will define specific values for registration types. See Section 7 for more information.

The registrar SHOULD include a REG_RESPONSE parameter in its R2 or UPDATE packet only if a registration has successfully completed.

The registrar MUST NOT include more than one REG_RESPONSE parameter in its R2 or UPDATE packets, while the requester MUST be able to process one or more REG_RESPONSE parameters in received R2 or UPDATE packets.

The requester MUST be prepared to receive any registration lifetime, including ones beyond the minimum and maximum lifetime indicated in the REG_INFO parameter. It MUST NOT expect that the returned lifetime will be the requested one, even when the requested lifetime falls within the announced minimum and maximum.

HIP_SIGNATURE protects the parameter within the R2 and UPDATE packets.

4.5. REG_FAILED

```
    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              |             Length            |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   | Failure Type  |  Reg Type #1  |  Reg Type #2  |  Reg Type #3  |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
   |      ...      |     ...       |  Reg Type #n  |               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+    Padding    +
   |                                                               |
   +-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+-+
```

Type	936
Length	Length in octets, excluding Type, Length, and Padding.
Failure Type	Reason for failure.
Reg Type	The registration types that failed with the specified reason.

Value	Registration Failure Type
0	Registration requires additional credentials
1	Registration type unavailable
2	Insufficient resources
3	Invalid certificate
9-200	Unassigned
201-255	Reserved for Private Use

Other documents will define specific values for registration types. See Section 7 for more information.

Failure Type 0 (zero) indicates that the registrar requires additional credentials to authorize a requester to register with the registration types listed in the parameter. Failure Type 1 (one) indicates that the requested service type is unavailable at the registrar. Failure Type 2 indicates that the registrar does not currently have enough resources to register the requester for the service(s); when that is the case, the requester MUST NOT reattempt immediately to register for the same service(s) and MAY attempt to contact another registrar to register for the service(s). Failure Type 3 indicates that the registrar could not validate the certificate provided by the requester to register for the service(s); when that is the case, the requester MUST NOT reattempt to register for the same set of services while providing the same certificate and MAY attempt to register for the same set of services with a different certificate, or with a different set of services with the same certificate.

The registrar SHOULD include a REG_FAILED parameter in its R2 or UPDATE packet, if registration with the registration types listed has not completed successfully, and a requester is asked to try again with additional credentials.

HIP_SIGNATURE protects the parameter within the R2 and UPDATE packets.

5. Establishing and Maintaining Registrations

Establishing and/or maintaining a registration may require additional information not available in the transmitted REG_REQUEST or REG_RESPONSE parameters. Therefore, registration type definitions MAY define dependencies for HIP parameters that are not defined in this document. Their semantics are subject to the specific registration type specifications.

The minimum lifetime both registrars and requesters MUST support is 10 seconds, while they SHOULD support a maximum lifetime of 120 seconds, at least. These values define a baseline for the specification of services based on the registration system. They were chosen to be neither too short nor too long, and to accommodate for existing timeouts of state established in middleboxes (e.g., NATs and firewalls.)

A zero lifetime is reserved for canceling purposes. Requesting a zero lifetime for a registration type is equal to canceling the registration of that type. A requester MAY cancel a registration before it expires by sending a REG_REQ to the registrar with a zero lifetime. A registrar SHOULD respond and grant a registration with a zero lifetime. A registrar (and an attached service) MAY cancel a registration before it expires, at its own discretion. However, if it does so, it SHOULD send a REG_RESPONSE with a zero lifetime to all registered requesters.

6. Security Considerations

This section discusses the threats on the HIP registration protocol and their implications on the overall security of HIP. In particular, it argues that the extensions described in this document do not introduce additional threats to HIP.

The extensions described in this document rely on the HIP base exchange and do not modify its security characteristics, e.g., digital signatures or Hashed Message Authentication Code (HMAC). Hence, the only threat introduced by these extensions is related to the creation of soft registration state at the registrar.

Registrars act on a voluntary basis and are willing to accept being a Responder and then to create HIP associations with a number of potentially unknown hosts. Because they have to store HIP association state anyway, adding a certain amount of time-limited HIP registration states should not introduce any serious additional threats, especially because HIP registrars may cancel registrations at any time at their own discretion, e.g., because of resource constraints during an attack.

7. IANA Considerations

This section is to be interpreted according to "Guidelines for Writing an IANA Considerations Section in RFCs" [RFC5226].

[RFC5203], obsoleted by this document, made the following definitions and reservations in the "Parameter Types" subregistry under "Host Identity Protocol (HIP) Parameters":

Value	Parameter Type	Length
930	REG_INFO	variable
932	REG_REQUEST	variable
934	REG_RESPONSE	variable
936	REG_FAILED	variable

In the "Parameter Types" subregistry under "Host Identity Protocol (HIP) Parameters", the references to the obsoleted [RFC5203] have been replaced with references to this document.

[RFC5203], obsoleted by this document, requested the opening of the "Registration Types" subregistry under "Host Identity Protocol (HIP) Parameters", defined no registration types, but made the following reservations in that subregistry:

Reg Type	Service
201-255	Reserved by IANA for private use

Adding a new type requires new IETF specifications.

In the "Registration Types" subregistry under "Host Identity Protocol (HIP) Parameters", references to the obsoleted [RFC5203] have been replaced with references to this document.

[RFC5203], obsoleted by this document, requested the opening of the "Registration Failure Types" subregistry under "Host Identity Protocol (HIP) Parameters" and made the following definitions and reservations in that subregistry:

Failure Type	Reason
0	Registration requires additional credentials
1	Registration type unavailable
201-255	Reserved by IANA for private use

Adding a new type requires new IETF specifications.

In the "Registration Failure Types" subregistry under "Host Identity Protocol (HIP) Parameters", references to the obsoleted [RFC5203] have been replaced with references to this document, and the following HIP Registration Failure Types have been added:

Value	Registration Failure Type
2	Insufficient resources
3	Invalid certificate
4	Bad certificate
5	Unsupported certificate
6	Certificate expired
7	Certificate other
8	Unknown CA
201-255	Reserved for Private Use

8. References

8.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <http://www.rfc-editor.org/info/rfc2119>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7401] Moskowitz, R., Ed., Heer, T., Jokela, P., and T. Henderson, "Host Identity Protocol Version 2 (HIPv2)", RFC 7401, DOI 10.17487/RFC7401, April 2015, <http://www.rfc-editor.org/info/rfc7401>.

[RFC8002] Heer, T. and S. Varjonen, "Host Identity Protocol Certificates", RFC 8002, DOI 10.17487/RFC8002, October 2016, <http://www.rfc-editor.org/info/rfc8002>.

[RFC8004] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extension", RFC 8004, DOI 10.17487/RFC8004, October 2016, <http://www.rfc-editor.org/info/rfc8004>.

8.2. Informative References

[HIP-ARCH] Moskowitz, R. and M. Komu, "Host Identity Protocol Architecture", Work in Progress, draft-ietf-hip-rfc4423-bis-14, June 2016.

[HIP-NAT] Keranen, A., Melen, J., and M. Komu, "Native NAT Traversal Mode for the Host Identity Protocol", Work in Progress, draft-ietf-hip-native-nat-traversal-13, July 2016.

[RFC3234] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, DOI 10.17487/RFC3234, February 2002, <http://www.rfc-editor.org/info/rfc3234>.

[RFC5203] Laganier, J., Koponen, T., and L. Eggert, "Host Identity Protocol (HIP) Registration Extension", RFC 5203, DOI 10.17487/RFC5203, April 2008, <http://www.rfc-editor.org/info/rfc5203>.

Appendix A. Changes from RFC 5203

- Updated references to revised HIP specifications.
- Added a new registration Failure Type for use in case of insufficient resources available at the HIP registrar.
- Added requester authorization based on certificates and new registration Failure Types for invalid certificates.

Acknowledgments

The following people (in alphabetical order) have provided thoughtful and helpful discussions and/or suggestions that have helped to improve this document: Jeffrey Ahrenholz, Miriam Esteban, Ari Keranen, Mika Kousa, Pekka Nikander, and Hannes Tschofenig.

Lars Eggert has received funding from the European Union’s Horizon 2020 research and innovation program 2014-2018 under grant agreement No. 644866. This document reflects only the authors’ views, and the European Commission is not responsible for any use that may be made of the information it contains.

Ari Keranen suggested inclusion of the text specifying requester authorization based on certificates as a direct adaption of text found in the HIP native NAT traversal specification [HIP-NAT].

Thanks to Joel M. Halpern for performing the Gen-ART review of this document as part of the publication process.

Contributors

Teemu Koponen coauthored an earlier, experimental version of this specification [RFC5203].

Authors' Addresses

Julien Laganier
Luminate Wireless, Inc.
Cupertino, CA
United States of America

Email: julien.ietf@gmail.com

Lars Eggert
NetApp
Sonnenallee 1
Kirchheim 85551
Germany

Phone: +49 151 12055791
Email: lars@netapp.com
URI: http://eggert.org