J. Laganier
LIP / Sun Microsystems
T. Koponen
HIIT
L. Eggert
NEC

Internet-Draft
Expires: August 12, 2005

February 11, 2005

Host Identity Protocol (HIP) Registration Extension
draft-koponen-hip-registration-00

Status of this Memo

Copyright Notice

Abstract

This document specifies a registration mechanism for the Host Identity Protocol that allows hosts to register with services.

Table of Contents

1. Introduction

This document specifies an extension to the Host Identity Protocol (HIP) [1]. The extension provides a generic means for a host to register with a service. The service may be, for example, a HIP rendezvous server [4] or a middlebox [5].

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 [2].

2. Terminology

This section defines terminology used throughout the reminder of this document. Please note that common terminology is defined elsewhere [1].

Requester:
: a host registering to a registrar and thus requesting access to a service.

Registrar:
: an entity with which requesters register. A registrar is a logical part of a service, but it may also be an independent entity and shared by a number of services.

Service:
: a facility that amends the HIP capabilities or functionalities of its requesters. Examples include firewalls that support HIP

traversal or rendezvous servers.

Registration:
state stored by a requester, a registrar, and a service that indicates the relationship the requester and service have. A registration is soft state; it has an associated finite lifetime. Requesters can extend established registrations through refresh operations (re-registration).

Registration Type:
an identifier that is transformable to a definition of a service. For example, a rendezvous registration type transforms to a rendezvous service. The registration type provides the means for

a registrar to inform requesters about the services it represents, whereas requesters use registration types to indicate the services they wish register with.

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 its 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, the additional parameters defined in the remainder of this document are used to register with the service.

If registering begins with the HIP base exchange, the differences to the standard HIP base exchange [3] are as follows:

1. A host that is capable and willing to act as a registrar includes a REG_INFO parameter in the R1 packets it sends during base exchanges.

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

3. At this point, the registrar tries to authenticate the requester based on currently available information. The details of this

authentication procedure are not specified in this document. If the requester is authorized to register for the requested service(s), the registrar includes a corresponding REG_RESPONSE parameter in its R2 response. If the requester is not authorized, the registrar MUST NOT include the REG_RESPONSE parameter. However, if the registrar needs further authorization, it includes a REG_FAILED parameter with a failure type of zero in its R2 response instead of a REG_RESPONSE parameter.

4. If the registrar required further authorization and the requester has more credentials to pass, the requester tries to register with the service again after the HIP association has been established.

If the requester reuses an existing HIP association with the registrar, registering with a service occurs as follows:

1. A host that is capable and willing to act as a registrar includes a REG_INFO parameter in the R1 packets it sends during base

exchanges or later announces its capabilities by sending the parameter in an UPDATE packet.

2. A requester constructs and includes a corresponding REG_REQUEST in an UPDATE packet and sends it.

3. The registrar tries to to authenticate requester based on then available information, as above. If the requester is authorized to register for the requested service(s), the registrar includes a corresponding REG_RESPONSE parameter in its UPDATE response. If the registrar needs further authorization, the registrar includes a REG_FAILED parameter with a failure type of zero in its UPDATE response.

4. If the registrar required further authorization and the requester has more credentials to pass, the requester tries to register with the service again with the additional credentials.

If the requester has no HIP association established with the registrar, it SHOULD send the REG_REQUEST already in the I2 packet. This is to minimize the number of packets exchanged with the

registrar. A registrar MAY drop 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 TBD.

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 the created registration before its expiration. The requester may also register to new services and refresh existing registrations by re-registering with the services. These operations occur through a REG_REQUEST/REG_RESPONSE parameter exchange carried in a pair of UPDATE packets.

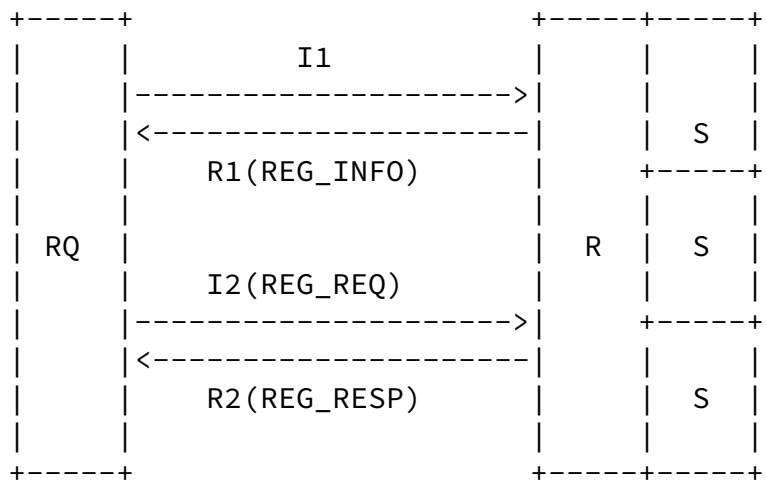
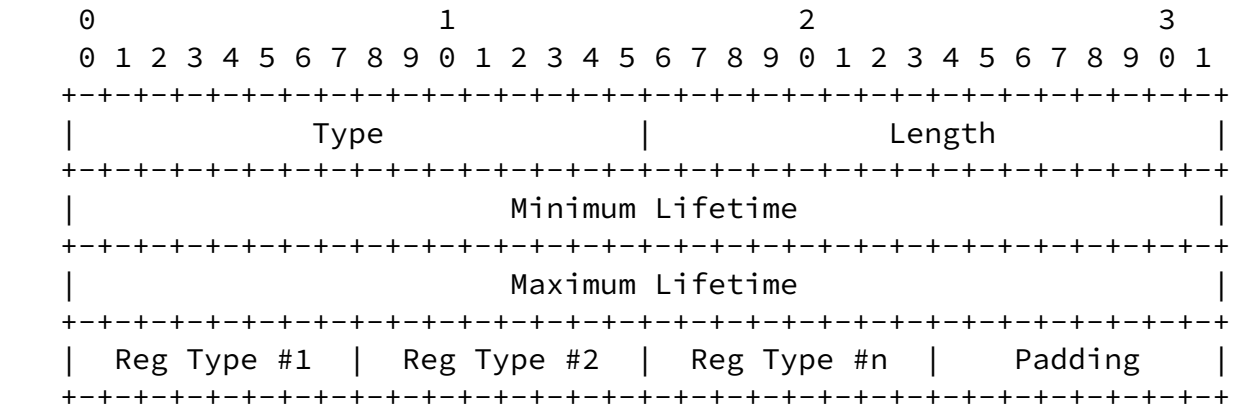


Figure 1: A requester (RQ) registers to a registrar (R) of services (S).

4. Parameter Formats and Processing

4.1 REG_INFO



Type	100 (for testing purposes until IANA assigns a number)
Length	Length in octets, excluding Type, Length, and Padding.
Min Lifetime	The minimum registration validity time (in seconds).
Max Lifetime	The maximum registration validity time (in seconds).
Reg Type	The registration types offered by the registrar.

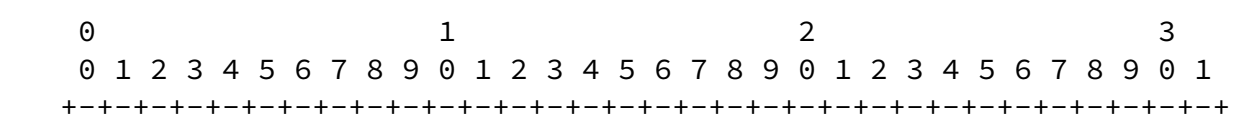
Other documents will define specific values for registration types.

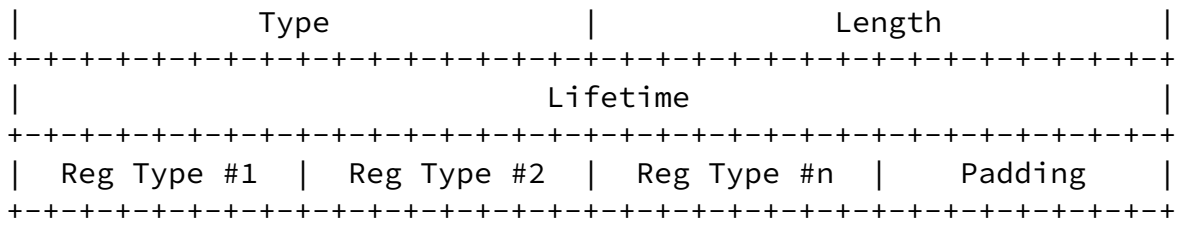
0-200	Reserved by IANA
201-255	Reserved by IANA for private use

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.

Signature protects the parameter within the R1 packets.

4.2 REG_REQUEST





Type 102 (for testing purposes until IANA assigns a number)
Length Length in octets, excluding Type, Length, and Padding.
Lifetime The registration validity time (in seconds).
Reg Type The preferred registration types in order of preference.

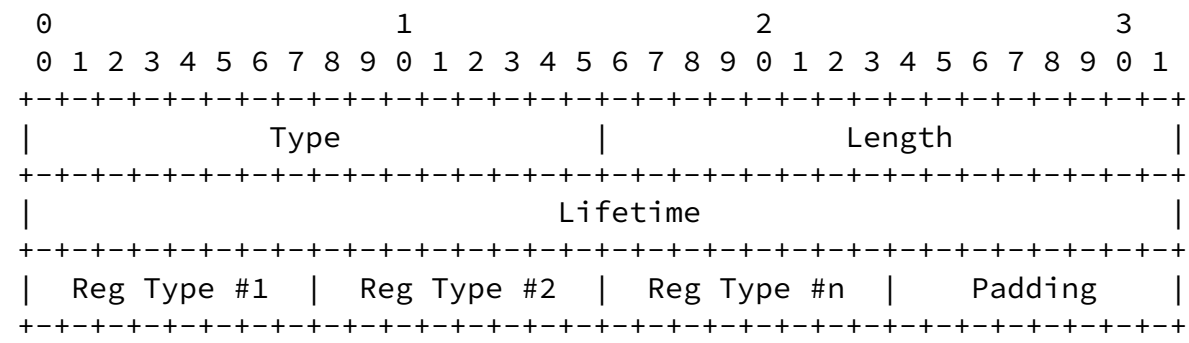
Other documents will define specific values for registration types.

0-200 Reserved by IANA
201-255 Reserved by IANA for private use

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 SHOULD NOT modify the registrations of registration types which are not listed in the parameter. Moreover, the requester SHOULD NOT include the parameter unless the registrar's I1 packet or an earlier received UPDATE packet has contained a REG_INFO parameter with the requested registration types.

The REG_REQUEST parameter in the I2 packet MUST be protected by a signature. The requester SHOULD support inclusion of multiple instances of the REG_REQUEST parameter in its I2 packets.

4.3 REG_RESPONSE



Type	104 (for testing purposes until IANA assigns a number)
Length	Length in octets, excluding Type, Length, and Padding.
Lifetime	The granted registration validity time (in seconds).
Reg Type	The granted registration types in order of preference.

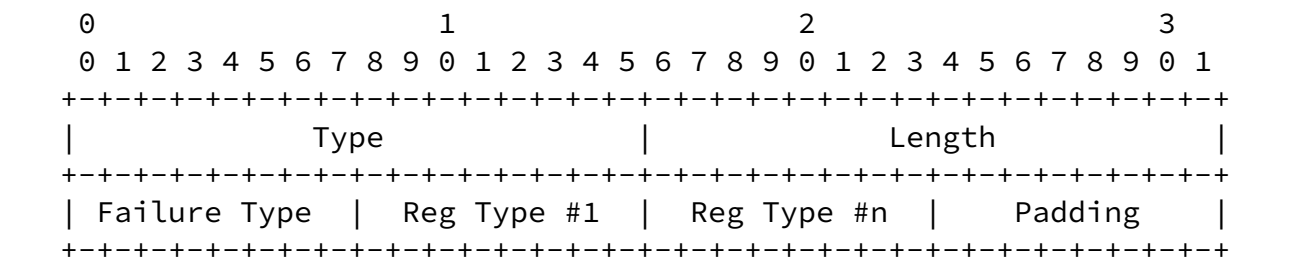
Other documents will define specific values for registration types.

0-200 Reserved by IANA
201-255 Reserved by IANA for private use

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

The registrar SHOULD be able to process and reply with separate REG_RESPONSE parameters to multiple instances of the REG_REQUEST parameters in incoming I2 and UPDATE packets.

4.4 REG_FAILED



Type	106 (for testing purposes until IANA assigns a number)
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.

Other documents will define specific values for registration types.

0-200 Reserved by IANA
201-255 Reserved by IANA for private use

A failure type of zero means a registrar needs more credentials to authorize a requester to register with the registration types listed in the parameter. Other failure types than zero have not been defined.

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

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 is subject to the specific registration type specification.

The minimum lifetime both registrars and requesters MUST support is 10 seconds, while they SHOULD support a maximum lifetime of 120 seconds, at least. [Comment.1]

A zero lifetime is reserved for cancelling purposes. Requesting a zero lifetime for a registration type equals to cancelling 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. The requester SHOULD prepare itself to the cancelling as the registered service can cancel the registration before the requester receives and processes a REG_RESP parameter acknowlegding the cancellation. 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

The security aspects of the HIP registration protocol are currently being investigated.

7. IANA Considerations

IANA has assigned the HIP parameter type numbers TBD to the registration parameter types and the notification error type number TBD to REG_REQUIRED notification error.

IANA needs to open a new registry for registration types. No types are defined in this document.

8. Acknowledgments

The following people have provided thoughtful and helpful discussions and/or suggestions that have improved this document: Pekka Nikander, Hannes Tschofenig, and Mika Kousa.

Lars Eggert was supported by the Ambient Networks project, partially funded by the European Commission under its Sixth Framework Programme. This document is provided "as is" and without any express or implied warranties, including, without limitation, the implied warranties of fitness for a particular purpose. The views and conclusions contained herein are those of the authors and should not be interpreted as necessarily representing the official policies or

endorsements, either expressed or implied, of the Ambient Networks project or the European Commission.

9. References

9.1 Normative References

[1] Moskowitz, R., "Host Identity Protocol Architecture", draft-ietf-hip-arch-00 (work in progress), October 2004.

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

[3] Moskowitz, R., "Host Identity Protocol", draft-ietf-hip-base-01 (work in progress), October 2004.

9.2 Informative References

[4] Laganier, J. and L. Eggert, "Host Identity Protocol (HIP) Rendezvous Extensions", draft-ietf-hip-rvs-00 (work in progress), October 2004.

[5] Carpenter, B. and S. Brim, "Middleboxes: Taxonomy and Issues", RFC 3234, February 2002.

[6] Saltzer, J., "On the Naming and Binding of Network Destinations", RFC 1498, August 1993.

[7] Kent, S. and R. Atkinson, "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[8] Klensin, J., "Role of the Domain Name System (DNS)", RFC 3467, February 2003.

Editorial Comments

[Comment.1] LE: Shouldn't we have a MUST and SHOULD for minimum lifetime and a separate MUST and SHOULD for maximum lifetime here?

Authors' Addresses

Julien Laganier
Sun Labs (Sun Microsystems) & LIP (CNRS/INRIA/ENSL/UCBL)
180, Avenue de l'Europe
Saint Ismier CEDEX 38334
FR

Phone: +33 476 188 815
EMail: ju@sun.com
URI: http://research.sun.com/

Teemu Koponen
Helsinki Institute for Information Technology
Advanced Research Unit (ARU)
P.O. Box 9800
Helsinki FIN-02015-HUT
FI

Phone: +358 9 45 1
EMail: teemu.koponen@hiit.fi
URI: http://www.hiit.fi/

Lars Eggert
NEC Network Laboratories
Kurfuerstenanlage 36
Heidelberg 69115
DE

Phone: +49 6221 90511 43
Fax: +49 6221 90511 55
EMail: lars.eggert@netlab.nec.de
URI: http://www.netlab.nec.de/

Appendix A. Document Revision History

Revision	Comments
00	Initial version.

specification can be obtained from the IETF on-line IPR repository at http://www.ietf.org/ipr.

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

Disclaimer of Validity

Copyright Statement

Acknowledgment

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