DHC
Internet-Draft
Expires: November 22, 2008

M. Stapp
Cisco Systems, Inc.

May 21, 2008

DHCPv6 Bulk Leasequery
draft-ietf-dhc-dhcpv6-bulk-leasequery-01.txt

Status of this Memo

Copyright Notice

Abstract

The Dynamic Host Configuration Protocol for IPv6 (DHCPv6) has been extended with a Leasequery capability that allows a client to request information about DHCPv6 bindings. That mechanism is limited to queries for individual bindings. In some situations individual binding queries may not be efficient, or even possible. This document expands on the Leasequery protocol, adding new query types and allowing for bulk transfer of DHCPv6 binding data via TCP.

Table of Contents

1. Introduction

The DHCPv6 [1] protocol specifies a mechanism for the assignment of IPv6 address and configuration information to IPv6 nodes. IPv6 Prefix Delegation for DHCPv6 (PD) [2] specifies a mechanism for DHCPv6 delegation of IPv6 prefixes and related data. DHCPv6 servers maintain authoritative information including binding information for delegated IPv6 prefixes.

The client of a PD binding is typically a router, which then advertises the delegated prefix to locally-connected hosts. The delegated IPv6 prefix must be routeable in order to be useful. The actual DHCPv6 PD client may not be permitted to inject routes into the delegating network. In service-provider (SP) networks, for example, an edge router typically acts as a DHCPv6 relay agent, and this edge router often has the responsibility to maintain routes within the service-provider network for clients' PD bindings.

A DHCPv6 relay with this responsibility requires a means to recover binding information from the authoritative DHCPv6 server(s) in the event of replacement or reboot, in order to restore routeability to delegated prefixes. The relay may be a network device without adequate local storage to maintain the necessary binding-to-route data. A DHCPv6 Leasequery protocol [6] has been developed that allows queries for individual bindings from the authoritative DHCPv6 Server(s). The individual query mechanism is only useable when the target binding is known to the requestor, such as upon receipt of traffic. In the case of DHCPv6 Prefix Delegation, the PD binding data may need to be known before any traffic arrives from the client router. The DHCPv6 relay router may not be able to form individual queries in such cases.

This document extends the DHCPv6 Leasequery protocol to add support for queries that address these requirements. At the SP edge there may be many thousands of delegated prefixes per relay, so we specify the use of TCP [3] for efficiency of data transfer. We specify a new

DHCPv6 option, the Relay Identifier option, to support efficient recovery of all data associated with a specific relay agent; we also add a query-type for this purpose. We add query-types by network segment and by Remote-ID option value, to assist a relay that needs to recover a subset of its clients' bindings.

2. Terminology

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 [4].

DHCPv6 terminology is defined in [1]. DHCPv6 Leasequery terminology is defined in [6].

3. Protocol Overview

The Bulk Leasequery mechanism is modeled on the existing individual Leasequery protocol in [6]; most differences arise from the use of TCP. A Bulk Leasequery client opens a TCP connection to a DHCPv6 Server, using the DHCPv6 port 547. Note that this implies that the Leasequery client has server IP address(es) available via configuration or some other means, and that it has unicast IP reachability to the server. No relaying for bulk leasequery is specified.

After establishing a connection, the client sends a LEASEQUERY message containing a query-type and data about bindings it is interested in. The server uses the query-type and the data to identify any relevant bindings. In order to support some query-types, servers may have to maintain additional data structures or be able to locate bindings based on specific option data. The server replies with a LEASEQUERY-REPLY message, indicating the success or failure of the query. If the query was successful, the server includes the first client's binding data in the LEASEQUERY-REPLY message also. If more than one client's bindings are being returned, the server then transmits the additional client bindings in a series of LEASEQUERY-DATA messages. If the server has sent at least one client's bindings, it sends a LEASEQUERY-DONE message when it has finished sending its replies. The client may reuse the connection to

send additional queries. Each end of the TCP connection can be closed after all data has been sent.

This specification includes a new DHCPv6 option, the Relay-ID option. The option contains a DUID identifying a DHCPv6 relay agent. Relay agents can include this option in Relay-Forward messages they send. Servers can retain the Relay-ID and associate it with bindings made on behalf of the relay's clients. A relay can then recover binding information about downstream clients by using the Relay-ID in a LEASEQUERY message. The Relay-ID option is defined in Section 5.4.1.

Bulk Leasequery supports the queries by IPv6 address and by Client DUID as specified in RFC5007 [6]. The Bulk Leasequery protocol also adds several new queries. The new queries introduced here cannot be used effectively with the UDP Leasequery protocol. Requestors MUST NOT send these new query-types in RFC5007 [6] query messages.

Query by Relay Identifier - This query asks a server for the bindings associated with a specific relay; the relay is identified by a DUID carried in a Relay-ID option.

Query by Link Address - This query asks a server for the bindings on a particular network segment; the link is specified in the query's link-address field.

Query by Remote ID - This query asks a server for the bindings associated with a Relay Agent Remote-ID option [5] value.

4. Interaction Between UDP Leasequery and Bulk Leasequery

Bulk Leasequery can be seen as an extension of the existing UDP Leasequery protocol [6]. This section tries to clarify the relationship between the two protocols.

The query-types introduced in the UDP Leasequery protocol can be used in the Bulk Leasequery protocol. One change in behavior is permitted

when Bulk Leasequery is used. RFC5007 [6], in sections 4.1.2.5 and 4.3.3, specifies the use of a Client Link option in LEASEQUERY-REPLY messages in cases where multiple bindings were found. When Bulk Leasequery is used, this mechanism is not necessary: a server returning multiple bindings simply does so directly as specified in this document. The Client Link option MUST NOT appear in Bulk Leasequery replies.

Only LEASEQUERY, LEASEQUERY-REPLY, LEASEQUERY-DATA, and LEASEQUERY-DONE messages are allowed over the Bulk Leasequery connection. No other DHCPv6 messages are supported. The Bulk Leasequery connection is not an alternative DHCPv6 communication option for clients seeking DHCPv6 service.

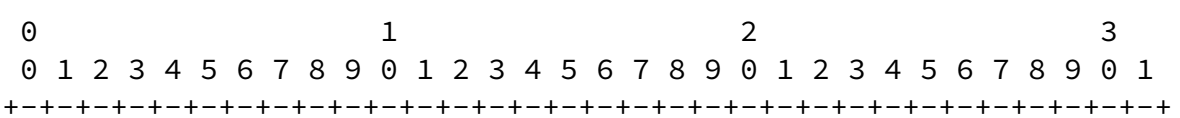
The new queries introduced in this specification cannot be used with the UDP Leasequery protocol. Servers that implement this specification and also permit UDP queries MUST NOT accept Bulk Leasequery query-types in UDP Leasequery messages. Such servers MUST respond with an error status code of NotAllowed [6].

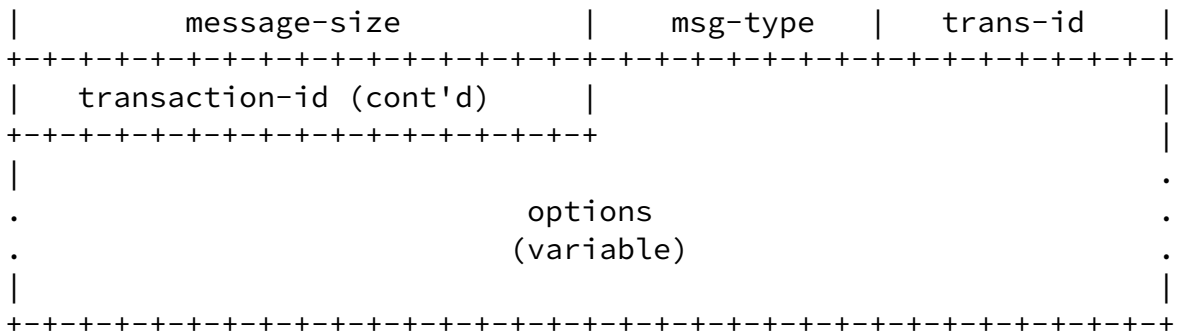
5. Message and Option Definitions

5.1. Message Framing for TCP

The use of TCP for the Bulk Leasequery protocol permits one or more DHCPv6 messages to be sent at a time. The receiver needs to be able to determine how large each message is. Two octets containing the message size in network byte-order are prepended to each DHCPv6 message sent on a Bulk Leasequery TCP connection. The two message-size octets 'frame' each DHCPv6 message.

DHCPv6 message framed for TCP:





message-size the number of octets in the message that follows, as a 16-bit integer in network byte-order.

All other fields are as specified in DHCPv6 [1].

5.2. Messages

The LEASEQUERY and LEASEQUERY-REPLY messages are defined in RFC5007 [6]. In a Bulk Leasequery exchange, a single LEASEQUERY-REPLY message is used to indicate the success or failure of a query, and to carry data that do not change in the context of a single query and answer, such as the Server-ID and Client-ID options. If a query is successful, only a single LEASEQUERY-REPLY message MUST appear. If the server is returning binding data, the LEASEQUERY-REPLY also contains the first client's binding data in an OPTION_CLIENT_DATA option.

5.2.1. LEASEQUERY-DATA

The LEASEQUERY-DATA message (message type TBD) carries data about a single DHCPv6 client's leases and/or PD bindings on a single link. The purpose of the message is to reduce redundant data when there are multiple bindings to be sent. The LEASEQUERY-DATA message MUST be preceded by a LEASEQUERY-REPLY message. The LEASEQUERY-REPLY conveys the query's status, carries the Leasequery's Client-ID and Server-ID

options, and carries the first client's binding data if the query was successful.

LEASEQUERY-DATA MUST ONLY be sent in response to a successful LEASEQUERY, and only if more than one client's data is to be sent. The LEASEQUERY-DATA message's transaction-id field MUST match the transaction-id of the LEASEQUERY request message. The Server-ID, Client-ID, and OPTION_STATUS_CODE options SHOULD NOT be included: that data should be constant for any one Bulk Leasequery reply, and should have been conveyed in the LEASEQUERY-REPLY message.

5.2.2. LEASEQUERY-DONE

The LEASEQUERY-DONE message (message type TBD) indicates the end of a group of related Leasequery replies. The LEASEQUERY-DONE message's transaction-id field MUST match the transaction-id of the LEASEQUERY request message. The presence of the message itself signals the end of a stream of reply messages. A single LEASEQUERY-DONE MUST BE sent after all replies (a successful LEASEQUERY-REPLY and zero or more LEASEQUERY-DATA messages) to a successful Bulk Leasequery request that returned at least one binding.

A server may encounter an error condition after it has sent the initial LEASEQUERY-REPLY. In that case, it SHOULD attempt to send a LEASEQUERY-DONE with an OPTION_STATUS_CODE option indicating the error condition to the requestor. Other DHCPv6 options SHOULD NOT be included in the LEASEQUERY-DONE message.

5.3. Query Types

The OPTION_LQ_QUERY option is defined in [6]. We introduce the following new query-types: QUERY_BY_RELAYID, QUERY_BY_LINK_ADDRESS, QUERY_BY_REMOTE_ID. These queries are designed to assist relay agents in recovering binding data in circumstances where some or all of the relay's binding data has been lost.

5.3.1. QUERY_BY_RELAYID

This query asks the server to return bindings associated with the specified relay DUID.

QUERY_BY_RELAYID (3) - The query-options MUST contain an OPTION_RELAYID option. If the link-address field is 0::0, the query asks for all bindings associated with the specified relay DUID. If the link-address is specified, the query asks for bindings on that link.

5.3.2. QUERY_BY_LINK_ADDRESS

The QUERY_BY_LINK_ADDRESS asks the server to return bindings on a network segment identified by an link-address value from a relay's Relay-Forward message.

QUERY_BY_LINK_ADDRESS (4) - The query's link-address contains an address a relay may have used in the link-address of a Relay-Forward message. The Server attempts to locate bindings on the same network segment as the link-address.

5.3.3. QUERY_BY_REMOTE_ID

The QUERY_BY_REMOTE_ID asks the server to return bindings associated with a Remote-ID option value from a relay's Relay-Forward message. The query-options MUST include a Relay Agent Remote-ID option [5].

In order to support this query, a server needs to record the most-recent Remote-ID option value seen in a Relay-Forward message along with its other binding data.

QUERY_BY_REMOTE_ID (5) - The query-options MUST include a Relay Agent Remote-ID option [5]. If the Server has recorded Remote-ID values with its bindings, it uses the option's value to identify bindings to return.

5.4. Options

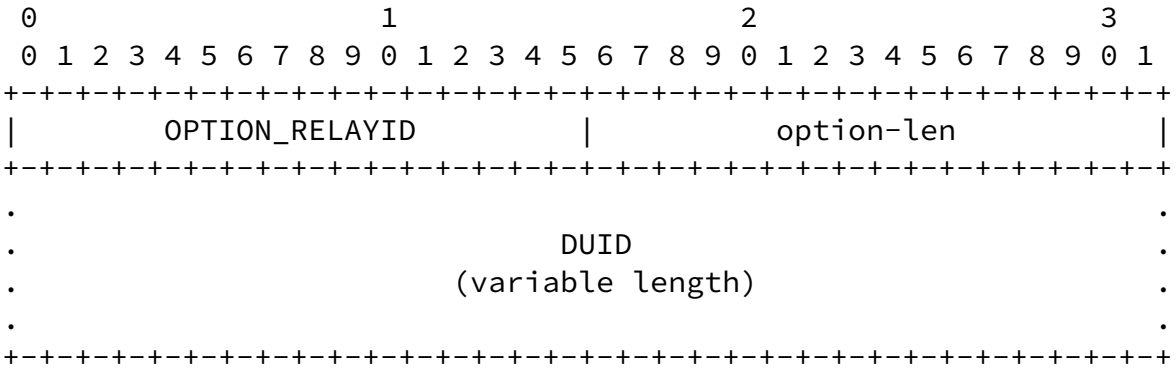
5.4.1. Relay-ID Option

The Relay-ID option carries a DUID [1]. A relay agent MAY include the option in Relay-Forward messages it sends. Obviously, it will not be possible for a server to respond to QUERY_BY_RELAYID queries unless the relay agent has included this option. A relay SHOULD be able to generate a DUID for this purpose, and capture the result in stable storage. A relay SHOULD also allow the DUID value to be configurable: doing so allows an administrator to replace a relay agent while retaining the association between the relay and existing DHCPv6 bindings.

A DHCPv6 Server MAY associate Relay-ID options from Relay-Forward

messages it processes with PD and/or lease bindings that result. Doing so allows it to respond to QUERY_BY_RELAYID Leasequeries.

The format of the Relay-ID option is shown below:



option-code OPTION_RELAYID (TBD).

option-len Length of DUID in octets.

DUID The DUID for the relay agent.

5.5. Status Codes

QueryTerminated (TBD) - Indicates that the server is unable to perform a query or has prematurely terminated the query for some reason (which should be communicated in the text message). This may be because the server is short of resources or is being shut down. The requestor may retry the query at a later time. The requestor should wait at least a short interval before retrying. Note that while a server may simply prematurely close its end of the connection, it is preferable for the server to send a LEASEQUERY-REPLY or LEASEQUERY-DONE with this status-code to notify the requestor of the condition.

5.6. Connection and Transmission Parameters

DHCPv6 Servers that support Bulk Leasequery SHOULD listen for incoming TCP connections on the DHCPv6 server port 547. Implementations MAY offer to make the incoming port configurable, but port 547 MUST be the default. Client implementations SHOULD make TCP connections to port 547, and MAY offer to make the destination server port configurable.

This section presents a table of values used to control Bulk Leasequery behavior, including recommended defaults. Implementations

MAY make these values configurable.

Parameter	Default	Description
BULK_LQ_CONN_TIMEOUT	30 secs	Bulk Leasequery connection timeout
BULK_LQ_DATA_TIMEOUT	30 secs	Bulk Leasequery data timeout
BULK_LQ_MAX_RETRY	60 secs	Max Bulk Leasequery retry timeout value
BULK_LQ_MAX_CONNS	10	Max Bulk Leasequery TCP connections

6. Requestor Behavior

6.1. Connecting

A Requestor attempts to establish a TCP connection to a DHCPv6 Server in order to initiate a Leasequery exchange. The Requestor SHOULD be prepared to abandon the connection attempt after BULK_LQ_CONN_TIMEOUT. If the attempt fails, the Requestor MAY retry. Retries MUST use an exponential backoff timer, increasing the interval between attempts up to BULK_LQ_MAX_RETRY.

6.2. Forming Queries

After a connection is established, the Requestor constructs a Leasequery message, as specified in [6]. The query may have any of the defined query-types, and includes the options and data required by the query-type chosen. The Requestor sends the message size then sends the actual DHCPv6 message, as described in Section 5.1.

If the TCP connection becomes blocked while the Requestor is sending its query, the Requestor SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT. We make this recommendation to allow Requestors to control the period of time they are willing to wait before abandoning a connection, independent of notifications from the TCP implementations they may be using.

6.3. Processing Replies

The Requestor attempts to read a LEASEQUERY-REPLY message from the

TCP connection. If the stream of replies becomes blocked, the Requestor SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT, and MAY begin retry processing if configured to do so.

The Requestor examines the LEASEQUERY-REPLY message, and determines how to proceed. Message validation rules are specified in DHCPv6 Leasequery [6]. If the reply contains an error status code (carried in an OPTION_STATUS_CODE option), the Requestor follows the

recommendations in [6]. A successful reply that does not include an OPTION_CLIENT_DATA option indicates that the target server had no bindings matching the query.

The Leasequery protocol uses the OPTION_CLIENT_LINK option as an indicator that multiple bindings were present in response to a single query. For Bulk Leasequery, the OPTION_CLIENT_LINK option is not used, and MUST NOT be present in replies.

A successful LEASEQUERY-REPLY that is returning binding data includes an OPTION_CLIENT_DATA option and possibly additional options. If there are additional bindings to be returned, they will be carried in LEASEQUERY-DATA messages. Each LEASEQUERY-DATA message contains an OPTION_CLIENT_DATA option, and possibly other options. A LEASEQUERY-DATA message that does not contain an OPTION_CLIENT_DATA MUST BE discarded.

A single bulk query can result in a large number of replies. For example, a single relay agent might be responsible for routes for thousands of clients' delegated prefixes. The Requestor MUST be prepared to receive more than one LEASEQUERY-DATA with transaction-ids matching a single LEASEQUERY message.

The LEASEQUERY-DONE message ends a successful Bulk Leasequery request that returned at least one binding. A LEASEQUERY-REPLY without any bindings MUST NOT be followed by a LEASEQUERY-DONE message for the same transaction-id. After receiving LEASEQUERY-DONE from a server, the Requestor MAY close the TCP connection to that server. If the transaction-id in the LEASEQUERY-DONE does not match an outstanding LEASEQUERY message, the client MUST close the TCP connection.

6.4. Querying Multiple Servers

A Bulk Leasequery client MAY be configured to attempt to connect to and query from multiple DHCPv6 servers in parallel. The DHCPv6 Leasequery specification [6] includes a discussion about reconciling binding data received from multiple DHCPv6 servers.

6.5. Multiple Queries to a Single Server

Bulk Leasequery clients may need to make multiple queries in order to recover binding information. A Requestor MAY use a single connection to issue multiple queries. Each query MUST have a unique transaction id. A server MAY process more than one query at a time. A server that is willing to do so MAY interleave replies to the multiple queries within the stream of reply messages it sends. Clients need to be aware that replies for multiple queries may be interleaved within the stream of reply messages. Clients that are not able to

process interleaved replies (based on transaction id) MUST NOT send more than one query at a time. Requestors should be aware that servers are not required to process queries in parallel, and that servers are likely to limit the rate at which they process queries from any one Requestor.

6.5.1. Example

This example illustrates what a series of queries and responses might look like. This is only an example - there is no requirement that this sequence must be followed, or that clients or servers must support parallel queries.

In the example session, the client sends four queries after establishing a connection. Query 1 results in a failure; query 2 succeeds and the stream of replies concludes before the client issues any new query. Query 3 and query 4 overlap, and the server interleaves its replies to those two queries.

```
     Client                          Server
     ------                          ------
     LEASEQUERY xid 1 ----->
                      <-----         LEASEQUERY-REPLY xid 1 (w/error)
     LEASEQUERY xid 2 ----->
                      <-----         LEASEQUERY-REPLY xid 2
```

```
                        <-----       LEASEQUERY-DATA xid 2
                        <-----       LEASEQUERY-DATA xid 2
                        <-----       LEASEQUERY-DONE xid 2
        LEASEQUERY xid 3 ----->
        LEASEQUERY xid 4 ----->
                        <-----       LEASEQUERY-REPLY xid 4
                        <-----       LEASEQUERY-DATA xid 4
                        <-----       LEASEQUERY-REPLY xid 3
                        <-----       LEASEQUERY-DATA xid 4
                        <-----       LEASEQUERY-DATA xid 3
                        <-----       LEASEQUERY-DONE xid 3
                        <-----       LEASEQUERY-DATA xid 4
                        <-----       LEASEQUERY-DONE xid 4
```

6.6. Closing Connections

The Requestor MAY close its end of the TCP connection after sending a LEASEQUERY message to the server. The Requestor MAY choose to retain the connection if it intends to issue additional queries. Note that this client behavior does not guarantee that the connection will be available for additional queries: the server might decide to close the connection based on its own configuration.

7. Server Behavior

7.1. Accepting Connections

Servers that implement DHCPv6 Bulk Leasequery listen for incoming TCP connections. Port numbers are discussed in Section 5.6. Servers MUST be able to limit the number of currently accepted and active connections. The value BULK_LQ_MAX_CONNS MUST be the default; implementations MAY permit the value to be configurable.

Servers MAY restrict Bulk Leasequery connections and LEASEQUERY messages to certain clients. Connections not from permitted clients SHOULD BE closed immediately, to avoid server connection resource exhaustion. Servers MAY restrict some clients to certain query types. Servers MAY reply to queries that are not permitted with the NotAllowed status code [6], or MAY close the connection.

If the TCP connection becomes blocked while the server is accepting a

connection or reading a query, it SHOULD be prepared to terminate the connection after BULK_LQ_DATA_TIMEOUT. We make this recommendation to allow Servers to control the period of time they are willing to wait before abandoning an inactive connection, independent of the TCP implementations they may be using.

7.2. Forming Replies

The DHCPv6 Leasequery [6] specification describes the initial construction of LEASEQUERY-REPLY messages and the processing of QUERY_BY_ADDRESS and QUERY_BY_CLIENTID. Use of the LEASEQUERY-REPLY and LEASEQUERY-DATA messages to carry multiple bindings are described in Section 5.2. Message transmission and framing for TCP is described in Section 5.1. If the connection becomes blocked while the server is attempting to send reply messages, the server SHOULD be prepared to terminate the TCP connection after BULK_LQ_DATA_TIMEOUT.

If the server encounters an error during initial query processing, before any reply has been sent, it SHOULD send a LEASEQUERY-REPLY containing an error code in an OPTION_STATUS_CODE option. This signals to the requestor that no data will be returned. If the server encounters an error while processing a query that has already resulted in one or more reply messages, the server SHOULD send a LEASEQUERY-DONE message with an error status. The server SHOULD close its end of the connection as an indication that it was not able to complete query processing.

If the server does not find any bindings satisfying a query, it SHOULD send a LEASEQUERY-REPLY without an OPTION_STATUS_CODE option and without any OPTION_CLIENT_DATA option. Otherwise, the server

sends each binding's data in a reply message. The first reply message is a LEASEQUERY-REPLY. The binding data is carried in an OPTION_CLIENT_DATA option, as specified in [6] and extended below. The server returns subsequent bindings in LEASEQUERY-DATA messages, which can avoid redundant data (such as the requestor's Client-ID).

For QUERY_BY_RELAYID, the server locates each binding associated with the query's Relay-ID option value. In order to give a meaningful reply to a QUERY_BY_RELAYID, the server has to be able to maintain this association in its DHCPv6 binding data. If the query's link-address is not set to 0::0, the server only returns bindings on links

that could contain that address. If the link-address is not 0::0 and the server cannot find any matching links, the server SHOULD return the NotConfigured status in a LEASEQUERY-REPLY.

For QUERY_BY_LINK_ADDRESS, the server locates each binding associated with the link identified by the query's link-address value.

For QUERY_BY_REMOTE_ID, the server locates each binding associated with the query's Relay Remote-ID option value. In order to be able to give meaningful replies to this query, the server has to be able to maintain this association in its binding database. If the query message's link-address is not set to 0::0, the server only returns bindings on links that could contain that address. If the link-address is not 0::0 and the server cannot find any matching links, the server SHOULD return the NotConfigured status in a LEASEQUERY-REPLY.

The server sends the LEASEQUERY-DONE message as specified in Section 5.2.

7.3. Multiple or Parallel Queries

As discussed in Section 6.5, Requestors may want to leverage an existing connection if they need to make multiple queries. Servers MAY support reading and processing multiple queries from a single connection. A server MUST NOT read more query messages from a connection than it is prepared to process simultaneously.

This MAY be a feature that is administratively controlled. Servers that are able to process queries in parallel SHOULD offer configuration that limits the number of simultaneous queries permitted from any one Requestor, in order to control resource use if there are multiple Requestors seeking service.

7.4. Closing Connections

The server MAY close its end of the TCP connection after sending its last message (a LEASEQUERY-REPLY or a LEASEQUERY-DONE) in response to

a query. Alternatively, the server MAY retain the connection and wait for additional queries from the client. The server SHOULD be prepared to limit the number of connections it maintains, and SHOULD be prepared to close idle connections to enforce the limit.

The server MUST close its end of the TCP connection if it encounters an error sending data on the connection. The server MUST close its end of the TCP connection if it finds that it has to abort an in-process request. A server aborting an in-process request MAY attempt to signal that to its clients by using the QueryTerminated (Section 5.5) status code. If the server detects that the client end has been closed, the server MUST close its end of the connection after it has finished processing any outstanding requests from the client.

8. Security Considerations

The "Security Considerations" section of [1] details the general threats to DHCPv6. The DHCPv6 Leasequery specification [6] describes recommendations for the Leasequery protocol, especially with regard to relayed LEASEQUERY messages, mitigation of packet-flooding DOS attacks, restriction to trusted clients, and use of IPsec [7].

The use of TCP introduces some additional concerns. Attacks that attempt to exhaust the DHCPv6 server's available TCP connection resources, such as SYN flooding attacks, can compromise the ability of legitimate clients to receive service. Malicious clients who succeed in establishing connections, but who then send invalid queries, partial queries, or no queries at all also can exhaust a server's pool of available connections. We recommend that servers offer configuration to limit the sources of incoming connections, that they limit the number of accepted connections and the number of in-process queries from any one connection, and that they limit the period of time during which an idle connection will be left open.

9. IANA Considerations

IANA is requested to assign a new DHCPv6 Option Code in the registry maintained in http://www.iana.org/assignments/dhcpv6-parameters:

OPTION_RELAYID

IANA is requested to assign a new value in the registry of DHCPv6 Status Codes maintained in http://www.iana.org/assignments/dhcpv6-parameters:

QueryTerminated

IANA is requested to assign values for the following new DHCPv6 Message types in the registry maintained in http://www.iana.org/assignments/dhcpv6-parameters:

LEASEQUERY-DONE
LEASEQUERY-DATA

IANA is requested to assign the following new values in the registry of query-types for the DHCPv6 OPTION_LQ_QUERY option:
QUERY_BY_RELAYID
QUERY_BY_LINK_ADDRESS
QUERY_BY_REMOTE_ID

10. Acknowledgements

Many of the ideas in this document were originally proposed by Kim Kinnear, Richard Johnson, Hemant Singh, Ole Troan, and Bernie Volz. Further suggestions and improvements were made by participants in the DHC working group, including John Brzozowski, Marcus Goller, Ted Lemon, and Bud Millwood.

11. Modification History

12. References

12.1. Normative References

[1] Droms, R., Bound, J., Volz, B., Lemon, T., Perkins, C., and M. Carney, "Dynamic Host Configuration Protocol for IPv6 (DHCPv6)", RFC 3315, July 2003.

[2] Troan, O. and R. Droms, "IPv6 Prefix Options for Dynamic Host Configuration Protocol (DHCP) version 6", RFC 3633, December 2003.

[3] Duke, M., Braden, R., Eddy, W., and E. Blanton, "A Roadmap for

RFC 4614, September 2006.

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

[5] Volz, B., "Dynamic Host Configuration Protocol for IPv6 (DHCPv6) Relay Agent Remote-ID Option", RFC 4649, August 2006.

[6] Brzozowski, J., Kinnear, K., Volz, B., and S. Zeng, "DHCPv6 Leasequery", RFC 5007, September 2007.

12.2. Informative References

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

Author's Address

Mark Stapp
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

Phone: +1 978 936 0000
Email: mjs@cisco.com

Full Copyright Statement

Intellectual Property

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.

Acknowledgment

Funding for the RFC Editor function is provided by the IETF Administrative Support Activity (IASA).