DHC
Internet-Draft
Expires: December 18, 2006

J. Brzozowski
Comcast Cable
K. Kinnear
B. Volz
S. Zeng
Cisco Systems, Inc.
June 16, 2006

DHCPv6 Leasequery
<draft-brzozowski-dhc-dhcvp6-leasequery-00.txt>

Status of this Memo

By submitting this Internet-Draft, each author represents that any applicable patent or other IPR claims of which he or she is aware have been or will be disclosed, and any of which he or she becomes aware will be disclosed, in accordance with Section 6 of BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF), its areas, and its working groups. Note that other groups may also distribute working documents as Internet-Drafts.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

The list of current Internet-Drafts can be accessed at http://www.ietf.org/ietf/1id-abstracts.txt.

The list of Internet-Draft Shadow Directories can be accessed at http://www.ietf.org/shadow.html.

This Internet-Draft will expire on December 18, 2006.

Copyright Notice

Copyright (C) The Internet Society (2006).

Abstract

This document specifies leasequery for the Dynamic Host Configuration Protocol for IPv6 (DHCPv6) which can be used as a means to obtain lease information about DHCPv6 clients from a DHCPv6 server. This document specifies the scope of data that can be retrieved as well as both DHCPv6 leasequery requestor and server behavior.

Table of Contents

1. Introduction

The DHCPv6 [2] protocol specifies a mechanism for the assignment of both IPv6 address and configuration information to IPv6 nodes. IPv6 Prefix Options for DHCPv6 [4] specifies a mechanism for the automated delegation of IPv6 prefixes and related options. Similar to DHCPv4 [5], DHCPv6 servers maintain authoritative information related to its operations including but not limited to lease information for IPv6 addresses and delegated prefixes.

The requirement exists in various types of IPv6 deployments, particularly those of a broadband variety, to leverage DHCPv6 [2] for retrieving data related to the operation of DHCPv6 servers programmatically. In particular it is desirable to be able to extract lease information about IPv6 addresses and delegated prefixes assigned using DHCPv6 [2, 4]. Specific examples where this information has illustrated value are in broadband networks to facilitate access control by edge devices. This capability to programitcally extract lease data from the DHCPv6 server is called leasequery.

Existing specifications, such as [3] are leveraged as a basis for the definition of the DHCPv6 leasequery protocol. The motivations and justifications identified in [3] also generally apply to this specification. Furthermore, advancements in DHCPv6 [2] are expanded upon to specify additional means by which IPv6 address and delegated prefix lease data can be retrieved through DHCPv6 leasequery.

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 [1].

DHCPv6 terminology is defined in [2]. Terminology specific to DHCPv6 leasequery can be found below:

client(s) The nodes that have one or more bindings with a DHCPv6 server. This does not refer to the node issuing the LEASEQUERY unless it itself has one or more bindings with a DHCPv6 server.

requestor The node that sends LEASEQUERY messages to one or more servers to retrieve information on the bindings for a client or clients.

3. Protocol Overview

The focus of this document is to define a protocol that allows processes and devices that wish to access information from a DHCPv6 server to do so in a lightweight and convenient manner. It is especially appropriate for processes and devices that already interpret DHCPv6 messages.

One important motivating example is that the LEASEQUERY message allows access concentrators to query DHCP servers to obtain location information of broadband access network devices.

The LEASEQUERY message is a query message only and does not affect the state of the IPv6 address or prefix, or the binding information associated with it.

The leasequery capability described in this document parallels the DHCPv4 leasequery capability documented in [3]. As such, it shares many of the basic motivations, design goals and constraints as the capability described in Section 4 of [3]. While the DHCPv4 leasequery capability was structured to return only a relatively small amount of information on each request, some aspects of the DHCPv6 leasequery capability may require the return of considerably more data than will fit in a single DHCPv6 reply. The following sections describe these situations.

3.1. On-demand Leasequery

One approach for requesting information from a DHCPv6 server using the leasequery capability is to request just the information necessary to satisfy an immediate need. If the requestor is an access concentrator, then the immediate need will typically be that it has received an IPv6 packet and it needs to refresh its information concerning the DHCPv6 client to which that an IPv6 address is currently leased. In this case, the request will typically be by Address, DUID, or by Device ID. See Section 3.3 for details. The information returned will typically be for a single client. The information will fit in the return message, and so no special handling is required. This approach fits clearly into the single request/response cycle common to other DHCPv6 message exchanges.

3.2. Anticipatory Leasequery

A second approach for requesting information from a DHCPv6 server is to use the leasequery capability to rebuild an internal data store containing information available from a DHCPv6 server. The rebuilding of the data store in this approach takes place as soon as

possible after the need to rebuild it is discovered (such as on booting), and doesn't wait on the receipt of specific packets to trigger a piecemeal database update (as is the case for on-demand leasequery). In this approach, the requestor will send LEASEQUERY messages in advance of actually receiving any packets to which the leasequery information should relate.

The requests that a requestor would use to employ this approach would typically be those by Prefix or Link.

The challenge presented by this approach is that the quantity of information returned to the requestor is usually many times more than could fit into a single LEASEQUERY-REPLY message. Thus, to support the anticipatory leasequery approach we need to develop a way to return multiple reply messages that all relate to a single query.

When the reply to a LEASEQUERY message will not fit entirely within one LEASEQUERY-REPLY message, then an OPTION_LQ_COOKIE option is returned in the LEASEQUERY-REPLY message. The appearance of the OPTION_LQ_COOKIE option in a LEASEQUERY-REPLY message indicates to the requestor that additional data beyond that in the current LEASEQUERY-REPLY message is available for the query. The requestor then submits the LEASEQUERY message, with the same OPTION_LQ_QUERY option but with the addition of the OPTION_LQ_COOKIE option, to the DHCPv6 server. When the DHCPv6 server receives the LEASEQUERY message, the information in the OPTION_LQ_COOKIE option allows the DHCPv6 server to return additional information concerning the results of the query.

If you visualize the information that a DHCPv6 server would return from a single leasequery as a sequence of OPTION_CLIENT_DATA options, then the contents of the OPTION_LQ_COOKIE option are essentially a key into that sequence, and allow the server to return the next items in the sequence along with a different OPTION_LQ_COOKIE option with a new key into the sequence of OPTION_CLIENT_DATA options still to be returned.

See Section 4.5 for specific details of this approach.

3.3. Query Types

Leasquery provides for the following queries:

Query by DUID - This query allows a device to request from a server the bindings for a specific client on a specific link or any instances of the client on any of the server's configured links.

Query by IPv6 address - This query allows a device to request from a server the bindings for a client that either is bound to the address or has been delegated the prefix that contains the address. This is likely the on-demand request that an access concentrator that receives an IPv6 packet would make.

Query by IPv6 prefix - This query allows a device to request from a server the bindings for clients that have been assigned addresses or delegated prefixes within the requested prefix, or were delegated the prefix.

Query by Link - This query allows a device to request from a server the bindings for clients on a particular link or on all links configured in a server. This is likely the anticipatory request that an access concentrator would make when it discovers a need to rebuild its data store.

Query by Remote ID - This query allows a device to request from a server the clients and their bindings that match a remote-id.

Query by Device ID - This query allows a device to request from a server the clients and their bindings that match a device-id.

It is important to note that all queries determine the clients to be returned and that all bindings for a client (on a link) are returned. Thus, a Query by IPv6 address may return one client and will return all of the bindings that client has on the link for the address. It is up to the receiver of the information as to what information it wants to use.

4. Protocol Details

4.1. Message and Option Definitions

4.1.1. Messages

The LEASEQUERY and LEASEQUERY-REPLY messages use the Client/Server message formats described in [2], section 6. Two new message codes are defined:

LEASEQUERY (TBD) - A requestor sends a LEASEQUERY message to any available server to obtain information on a client's or clients' leases. The options in an OPTION_LQ_QUERY determine the query.

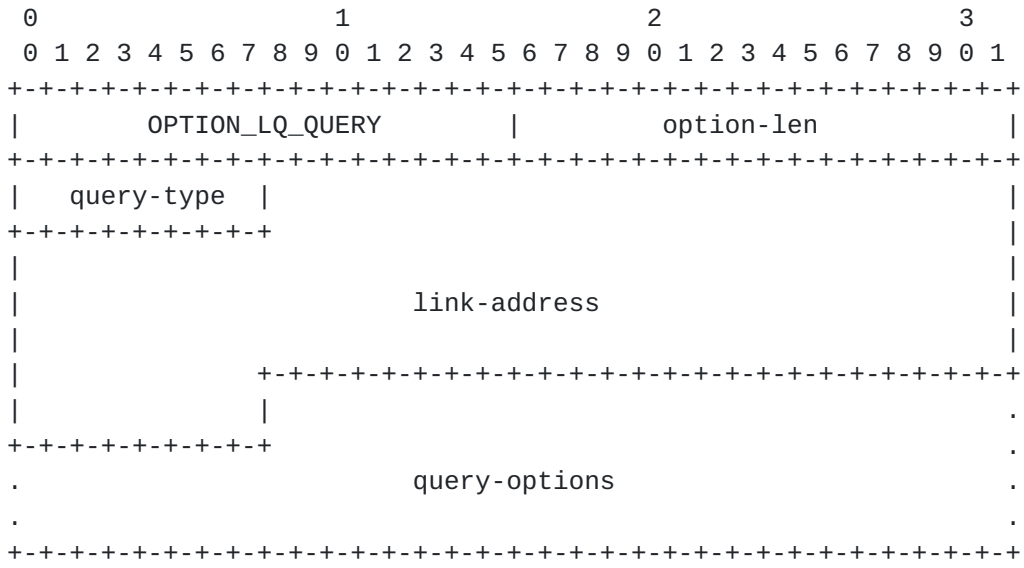
LEASEQUERY-REPLY (TBD) - A server sends a LEASEQUERY-REPLY message containing client data in response to a LEASEQUERY message.

4.1.2. Options

4.1.2.1. Query Option

The Leasequery Query option is used only in a LEASEQUERY message and identifies the query being performed. The option includes the query type, link-address (or 0::0 for all links), and option(s) to provide data needed for the query.

The format of the Query option is shown below:



option-code OPTION_LQ_QUERY (TBD)

option-len 17 + length of query-options field.

link-address A global address that will be used by the server to identify the link to which the query applies, or 0::0 if the server is to search all its links.

query-type the query requested (see below).

query-options the options related to the query.

The query-type and required query-options for the query types (Section 3.3) are:

QUERY_BY_CLIENTID (1) - The query-options MUST contain an OPTION_CLIENTID option [2]. The link-address field MUST specify an address for the link on which the client is located or 0::0 for all links for which the server is configured. Only the requested client's information is returned (if available) in the OPTION_CLIENT_DATA option. If the link-address is 0::0, multiple OPTION_CLIENT_DATA options may be returned, one for each link on which the client has one or more bindings.

QUERY_BY_ADDRESS (2) - The query-options MUST contain an OPTION_IAADDR option [2]. The link-address field MUST specify an address for the link on which the address is located if the address in the OPTION_IAADDR option is of insufficient scope or 0::0. Only the information for a client that has a lease for the specified address or was delegated a prefix that contains the specified address is returned (if available).

QUERY_BY_PREFIX (3) - The query-options MUST contain an OPTION_IAPREFIX option [4]. The link-address field MUST specify an address for the link on which the prefix is located if the prefix in the OPTION_IAPREFIX option is of insufficient scope or 0::0. Only the information for clients with bindings on the prefix or were delegated the specified prefix are returned (if available).

QUERY_BY_LINK (4) - There are no required query-options options for this query. The link-address field MUST specify an address for the link on which the clients are located or 0::0 for all links for which the server is configured. Only information for the clients that were assigned addresses or delegated prefixes on the specified link is returned (if available).

QUERY_BY_REMOTE_ID (5) - The query-options MUST contain an OPTION_REMOTE_ID option [7]. The link-address field MUST specify an address for the link on which the client is located or 0::0 for all links for which the server is configured. Only the clients on the link or on all links that have the specified remote-id are returned.

QUERY_BY_DEVICE_ID (6) - The query-options MUST contain an OPTION_DEVICE_ID [8]. The link-address field MUST specify an address for the link on which the client is located or 0::0 for all links for which the server is configured. Only the clients on the link or on all links that have the specified device-id are returned.

The query-options MAY also include an OPTION_ORO option [2] to

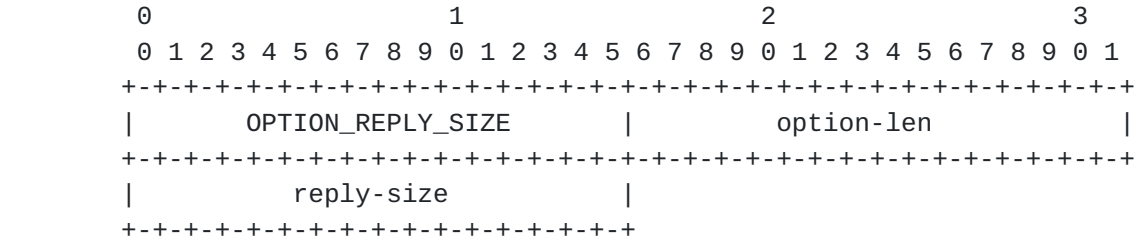
indicate the options for each client that the requestor would like the server to return. Note that this OPTION_ORO is distinct and separate from an OPTION_ORO that may be in the requestor's LEASEQUERY message.

If a server receives an OPTION_LQ_QUERY with a query-type it does not support, the server SHOULD return an UnknownQueryType status-code. If a server receives a supported query-type but the query-options is missing a required option, the server SHOULD return a MalformedQuery status-code.

4.1.2.2. Reply Size Option

The Reply Size option is sent to a server in a LEASEQUERY message to indicate the largest LEASEQUERY-REPLY message that the requestor is prepared to receive. A server uses the option to determine how much data it can return to the requestor in a single reply.

The format of the Reply Size option is shown below:



option-code OPTION_REPLY_SIZE (TBD)

option-len 2

reply-size The maximum number of octets the requestor is prepared to receive from the server. This limits the size of the LEASEQUERY-REPLY message.

If a requestor does not supply this option, the server MUST use 1024 octets. The requestor must be aware of the recommendations on packet sizes and the use of fragmentation in section 5 of [6].

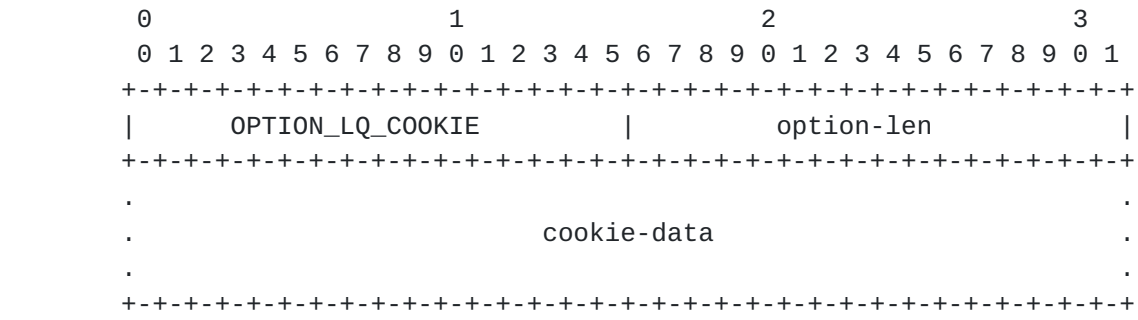
Given a reply size, if a LEASEQUERY-REPLY message can not contain at least one OPTION_CLIENT_DATA (and OPTION_LQ_COOKIE if more than one client's data is to be returned) the server must return the LEASEQUERY-REPLY with an OPTION_STATUS_CODE option [2] with the TooShort status-code.

4.1.2.3. Cookie Option

The Leasequery Cookie option is returned by a server when the reply can not fit into a single LEASEQUERY-REPLY message. The requestor may then send the same request, including the server's Cookie Option, to obtain the next portion of the reply. A server's reply is complete when no Cookie option is included.

If a requestor sends a request with this option, it MUST also include the OPTION_SERVERID option [2] that was returned in the LEASEQUERY-REPLY. A server MUST discard any request that includes this option without an OPTION_SERVERID option.

The format of the LEASEQUERY Cookie option is shown below:



option-code OPTION_LQ_COOKIE (TBD)

option-len length, in octets, of the cookie-data field. The minimum length is 1 octet.

cookie-data The opaque cookie data.

The cookie-data field's contents and length are completely up to the server and MUST be treated as opaque data by the requestor. The requestor MUST send the same option back to the server to retrieve the next portion of a reply. The cookie-data serves to identify to the server the next client to return.

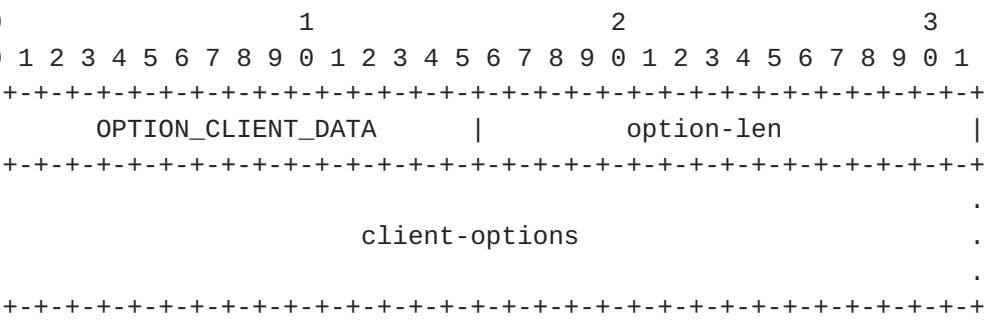
If the server receives this option with cookie-data that it can not validate, the server MUST return a LEASEQUERY-REPLY with an OPTION_STATUS_CODE option [2] with the StaleCookie status-code.

4.1.2.4. Client Data Option

The Client Data option is used to encapsulate the data for a single client on a single link in a LEASEQUERY-REPLY message. If a query is done for multiple links, multiple Client Data options for the same client may be returned and the addresses and/or delegated prefixes in

the binding for the client should be sufficient for the requestor to determine the link.

The format of the Client Data option is shown below:



option-code OPTION_CLIENT_DATA (TBD)

option-len length, in octets, of the encapsulated client-options field.

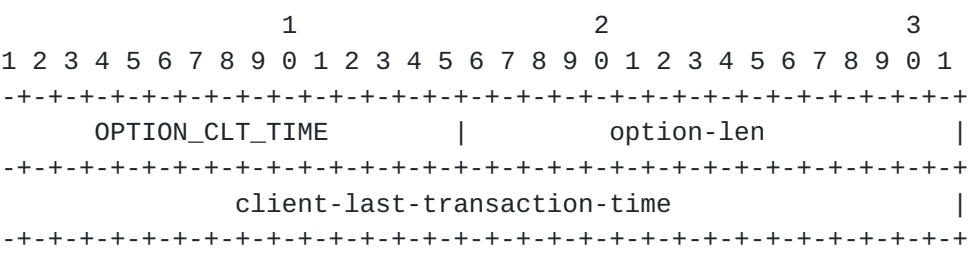
client-options the options associated with this client.

The encapsulated client-options include the OPTION_CLIENTID, OPTION_IAADDR, OPTION_IAPREFIX, and OPTION_CLT_TIME options and other options specific to the client and requested by the requestor in the OPTION_ORO in the OPTION_LQ_QUERY's query-options. The server MUST return all of the clients' statefully assigned addresses and delegated prefixes, with a non-zero valid lifetime, on the link.

4.1.2.5. Client Last Transaction Time Option

The Client Last Transaction Time option is encapsulated in an OPTION_CLIENT_DATA and identifies how long ago the server last communicated with the client, in seconds.

The format of the Client Last Transaction Time option is shown below:



option-code OPTION_CLT_TIME (TBD)

option-len 4

client-last-transaction-time
the number of seconds since the server last communicated with the client (on that link).

The client-last-transaction-time is a positive value and reflects the number of seconds since the server last communicated with the client (on that link).

4.1.3. Status Codes

The following new status codes are defined:

UnknownQueryType (TBD) - The query-type is unknown to or not supported by the server.

MalformedQuery (TBD) - The query is not valid, for example a required query-option is missing from the OPTION_LQ_QUERY.

StaleCookie (TBD) - The cookie supplied by the requestor in a LEASEQUERY message is not (or no longer) valid. The requestor SHOULD repeat the LEASEQUERY from the beginning (i.e., without a cookie). However, it SHOULD only retry a query a limited number of times.

TooShort (TBD) - The Reply Message Size option (or the default size) is too short to contain at least a single OPTION_CLIENT_DATA. The requestor should retry the query with a larger reply-size value. A simple policy is to use a small size until a larger is needed, and then use the largest desired to complete that query.

NotConfigured (TBD) - The server does not have the target address, prefix, or link in its configuration.

NotAllowed (TBD) - The server does not allow the requestor to issue this LEASEQUERY.

4.1.4. Transmission and Retransmission Parameters

This section presents a table of values used to describe the message transmission behavior for leasequery.

Parameter	Default	Description
LQ_TIMEOUT	1 sec	Initial LEASEQUERY timeout
LQ_MAX_RT	10 secs	Max LEASEQUERY timeout value
LQ_MAX_RC	5	Max LEASEQUERY retry attempts

4.2. Message Validation

4.2.1. LEASEQUERY

Requestors and clients MUST discard any received LEASEQUERY messages.

Servers MUST discard any received LEASEQUERY messages that meet any of the following conditions:

- the message does not include an OPTION_CLIENTID option.
- the message includes an OPTION_SERVERID option but the contents of the OPTION_SERVERID option does not match the server's identifier.
- the message does not include an OPTION_LQ_QUERY option.

4.2.2. LEASEQUERY-REPLY

Requestors MUST discard any received LEASEQUERY-REPLY messages that meet any of the following conditions:

- the message does not include an OPTION_SERVERID option.
- the message does not include an OPTION_CLIENTID option or the contents of the OPTION_CLIENTID option do not match the DUID of the requestor.
- the "transaction-id" field in the message does not match the value used in the original message.

Servers and Relay Agents (on the server port, 547 [2]) MUST discard any received LEASEQUERY-REPLY messages.

4.3. DHCPv6 Leasequery Requestor Behavior

This section describes how a requestor initiates lease data retrieval from DHCPv6 servers. If the server generates more data than can fit

into one reply message, then the requestor continues lease data retrieval as described in Section 4.5.

4.3.1. Creation of LEASEQUERY

The requestor sets the "msg-type" field to LEASEQUERY. The requestor generates a transaction ID and inserts this value in the "transaction-id" field.

The requestor MUST include an OPTION_CLIENTID option to identify itself to the server.

The requestor MUST include an OPTION_LQ_QUERY option and set the query-type, link-address, and query-options as appropriate to the query-type (Section 4.1.2.1).

The requestor SHOULD include an OPTION_SERVERID if it is not unicasting the LEASEQUERY yet only wants a response from a specific server.

The requestor SHOULD include an OPTION_REPLY_SIZE to specify the largest LEASEQUERY-REPLY it is willing to accept.

4.3.2. Transmission of LEASEQUERY

The requestor MAY be configured to use a list of destination addresses, which MAY include unicast addresses, the All_DHCP_Servers multicast address, or other addresses selected by the network administrator. If the requestor has not been explicitly configured, it MAY use the All_DHCP_Servers multicast address as the default.

The requestor SHOULD send LEASEQUERY to one or more DHCPv6 servers which are known to possess authoritative information concerning the query target.

In the absence of information concerning which DHCPv6 servers might possess authoritative information on the query target, the requestor SHOULD send LEASEQUERY to all DHCPv6 servers that the requestor knows about or is configured with. For example, the requestor MAY send LEASEQUERY to the All_DHCP_Servers multicast address.

The requestor transmits LEASEQUERY messages according to section 14 of [2], using the following parameters:

IRT	LQ_TIMEOUT
MRT	LQ_MAX_RT
MRC	LQ_MAX_RC
MRD	0

If the message exchange fails, the requestor takes an action based on the requestor's local policy. Examples of actions the requestor might take include:

- Select another server from a list of servers known to the requestor.
- Send to multiple servers by multicasting to the All_DHCP_Servers address.
- Terminate the leasequery.

4.3.3. Receipt of LEASEQUERY-REPLY

If a LEASEQUERY-REPLY contains an OPTION_LQ_COOKIE, the requestor SHOULD continue the bulked leasequery as described in Section 4.5. This section specifies the behavior of the requestor regarding the processing of a single (or initial) LEASEQUERY-REPLY message.

A successful LEASEQUERY-REPLY is one without an OPTION_STATUS_CODE with an error code and may or may not contain client data in OPTION_CLIENT_DATA options. A successful LEASEQUERY MAY contain no OPTION_CLIENT_DATA if no clients matched the query.

An unsuccessful LEASEQUERY-REPLY is one that has an OPTION_STATUS_CODE with an error code.

4.3.3.1. Receiving Successful LEASEQUERY-REPLY

Upon the receipt of a successful LEASEQUERY-REPLY in response to a LEASEQUERY, the requestor MUST extract the client data in the LEASEQUERY-REPLY and may update its binding information database.

The LEASEQUERY-REPLY SHOULD contain an OPTION_SERVER_RSN option [9] and the requestor SHOULD only update its binding information database as described in [9].

If an OPTION_CLIENT_DATA contains no OPTION_CLT_TIME, then the requestor SHOULD silently discard the OPTION_CLIENT_DATA option. The existence of such an OPTION_CLIENT_DATA SHOULD NOT affect the processing of other OPTION_CLIENT_OPTIONs by the requestor.

The requestor MUST be prepared to handle an OPTION_CLIENT_DATA that contains more or fewer options than listed in the OPTION_ORO of the LEASEQUERY message, and to handle multiple OPTION_CLIENT_DATA options, as many queries may return data on more than a single client.

4.3.3.2. Receiving Unsuccessful LEASEQUERY-REPLY

An unsuccessful LEASEQUERY-REPLY contains an OPTION_STATUS_CODE with one of the status codes listed in Section 4.1.3 or in [2] except Success.

Depending on the status code, the requestor may retry the query (such as for TooShort, but with a larger reply-size in the OPTION_REPLY_SIZE option, and for StaleCookie, but without the OPTION_LQ_COOKIE option), try a different server (such as for NotAllowed, NotConfigured, and UnknownQueryType), or try a different or corrected query (such as for UnknownQueryType and MalformedQuery).

4.3.4. Handling DHCPv6 Client Data from Multiple Sources

A requestor may receive lease data on the same client from the same DHCPv6 server in response to different types of LEASEQUERY. If a LEASEQUERY is sent to multiple servers, the requestor may receive from several servers lease data on the same DHCPv6 client. Additionally, if a requestor is an access concentrator, it may receive lease data from other than leasequery exchanges, e.g., [9]. This section describes how the requestor handles multiple lease data sources on the same DHCPv6 client from the same server or different servers.

The client data from the different sources may be disjoint or overlapping. The disjoint and overlapping relationship can happen between data from the same server or different servers.

If client data from two sources on the same client are of different types or values, then the data are disjoint. An example of data of different types is when a requestor receives an IPv6 address lease from one server and a prefix lease from another server, both assigned to the same client. An example of different values (but the same type) is when a requestor receives two IPv6 address leases from two different servers, both assigned to the same client, but the leases are on two different IPv6 addresses. If the requestor receives disjoint client data from different sources, it SHOULD merge them.

If client data from two sources on the same client are of the same type and value, then the data are overlapping. An example of overlapping data is when a requestor receives a lease on the same IPv6 address from two different servers. Overlapping client data are also called conflicting data.

The requestor SHOULD use the OPTION_SERVER_RSN [9] to resolve data conflicts originated from the same server, and SHOULD accept data with the higher server-sequence-number. The requestor SHOULD use the

OPTION_CLT_TIME to resolve data conflicts originated from different servers, and SHOULD accept data with most recent OPTION_CLT_TIME.

4.4. DHCPv6 Leasequery Server Behavior

A DHCPv6 server sends LEASEQUERY-REPLY messages in response to valid LEASEQUERY messages it receives to return the statefully assigned addresses, delegated prefixes, and other information about clients that match the query.

4.4.1. Receipt of LEASEQUERY Messages

Upon receipt of a valid LEASEQUERY message, the DHCPv6 server locates the requested clients, collects data on the clients, and constructs and returns a LEASEQUERY-REPLY. A LEASEQUERY message can not be used to assign, release, or otherwise modify bindings or other configuration information.

The server constructs a LEASEQUERY-REPLY message by setting the "msg-type" field to LEASEQUERY-REPLY, and copying the transaction ID from the LEASEQUERY message into the transaction-id field.

If the query-type in the OPTION_LQ_QUERY option is not a known or supported value, the server adds an OPTION_STATUS_CODE option with the UnknownQueryType status code and sends the LEASEQUERY-REPLY to the requestor. If the query-options do not contain the required options for the query-type, the server adds an OPTION_STATUS_CODE option with the MalformedQuery status code and sends the LEASEQUERY-REPLY to the client.

A server may also restrict LEASEQUERY messages, or query-types, to certain requestors. In this case, the server MAY discard the LEASEQUERY message or MAY add an OPTION_STATUS_CODE option with the NotAllowed status code and send the LEASEQUERY-REPLY to the requestor.

If the OPTION_LQ_COOKIE option is present in the LEASEQUERY message, the server is engaged in a bulked query with the requestor. The server behaviors upon the receipt of such a LEASEQUERY is described in Section 4.5.

If the OPTION_LQ_QUERY specified a non-zero link-address, the server MUST use the link-address to find the appropriate link for the client. Otherwise, the server uses the address from the OPTION_IAADDR option (if the query-type is QUERY_BY_ADDRESS) or the prefix from the OPTION_IAPREFIX option (if the query-type is QUERY_BY_PREFIX), to find the appropriate link for the client. There are three possible outcomes from this processing:

1. No address or prefix is specified, in which case the server MUST perform the query on all of its configured links.
2. The server locates the specified link, in which case the server MUST only perform the query on that link.
3. No link is found, in which case the server adds an OPTION_STATUS_CODE option with the NotConfigured status code and sends the LEASEQUERY-REPLY to the client.

At this point, the server uses the data in the OPTION_LQ_QUERY and OPTION_LQ_COOKIE (if present) to initiate or resume the query. The result of the query will be zero or more clients. This will result in zero or more OPTION_CLIENT_DATA option being added to the LEASEQUERY-REPLY and possibly a OPTION_LQ_COOKIE option as discussed in Section 4.4.3 and Section 4.5.

4.4.2. Processing a LEASEQUERY

The following sections assume a single link is being searched for the targets of a query. If multiple links are being searched, the order is up to the server and it essentially performs the actions below on each link. How and in what order a server performs the query is completely up to it; however, note the requirements in Section 4.5 to assure that a query eventually terminates. Requestors MUST NOT assume any specific behavior or order of the returned data.

4.4.2.1. QUERY_BY_DUID

This query requires that the server return the client identified by the OPTION_CLIENTID in the LEASEQUERY, if that client has a binding.

4.4.2.2. QUERY_BY_ADDRESS

This query requires that the server return the client that is bound to the address or that has been delegated the prefix that contains the address.

4.4.2.3. QUERY_BY_PREFIX

This query requires that the server return the clients that have bindings for addresses or delegated prefixes that are contained in the specified prefix. The server MUST only search its explicitly configured prefixes; this query type is not provided to allow a requestor to specify an arbitrary range of addresses about which to return bindings.

4.4.2.4. QUERY_BY_LINK

This query requires that the server return the clients that have

bindings for addresses or delegated prefixes on the specified link.

4.4.2.5. QUERY_BY_REMOTE_ID

This query requires that the server return the clients that match the specified OPTION_REMOTE_ID [7].

4.4.2.6. QUERY_BY_DEVICE_ID

This query requires that the server return the clients whose associated Device-IDs match the specified OPTION_DEVICE_ID in the LEASEQUERY [8].

4.4.3. Constructing a Client's OPTION_CLIENT_DATA

An OPTION_CLIENT_DATA option in a LEASEQUERY-REPLY message MUST minimally contain the following data.

1. OPTION_CLIENTID
2. OPTION_IAADDR
3. OPTION_IAPREFIX
4. OPTION_CLT_TIME

Depending on the bindings the client has on a link, either OPTION_IAADDR options, IAPREFIX options, or both may be present.

The OPTION_CLIENT_DATA SHOULD include options requested in the OPTION_ORO of the OPTION_LQ_QUERY option in the LEASEQUERY message and that are acceptable to return based on the list of "sensitive options", discussed below.

DHCPv6 servers SHOULD be configurable with a list of "sensitive options" that must not be returned to the requestor when specified in the OPTION_ORO of the OPTION_LQ_QUERY option in the LEASEQUERY message. Any option on this list MUST NOT be returned to a requestor, even if requested by that requestor.

4.4.4. Transmission of LEASEQUERY-REPLY Messages

The server sends the LEASEQUERY-REPLY message as described in the "Transmission of Reply Messages" section of [2].

4.5. Processing of Bulk Queries

Any time there is more data than will fit into a particular LEASEQUERY-REPLY message, then the DHCPv6 server MUST return a LEASEQUERY-REPLY message with an OPTION_LQ_COOKIE option indicating that there is more data available for this query.

All requestors MUST support bulked queries unless they restrict their queries to those that will only return a single client's data. For example, a Query By DUID with a specific link-address can ever only return a single client. A Query By Address should only return a single client; however that only holds if the DHCPv6 server is not performing stateful address assignment or prefix delegation for a delegated prefix.

Whenever a DHCPv6 server is preparing information for transmission to the requestor, it MUST ensure that all of the information in a particular OPTION_CLIENT_DATA option will fit in the LEASEQUERY-REPLY message. That is to say that the information in a single OPTION_CLIENT_DATA option MUST NOT be split over two LEASEQUERY-REPLY messages.

In the event that an OPTION_CLIENT_DATA will not fit entirely within a LEASEQUERY-REPLY message, the OPTION_CLIENT_DATA MUST NOT be placed in that message but instead an OPTION_LQ_COOKIE option MUST BE placed in the LEASEQUERY-REPLY message indicating that additional information is available from the DHCPv6 server regarding the current LEASEQUERY query. If there is not enough space, then the last OPTION_CLIENT_DATA option (or options) MUST be removed from the LEASEQUERY-REPLY message in order to make enough room for the OPTION_LQ_COOKIE option.

The information in the OPTION_LQ_COOKIE option is opaque to everyone but the DHCPv6 server who created the OPTION_LQ_COOKIE option. It is only used to tell the DHCPv6 server to continue returning additional data for this query and to give the DHCPv6 server some information as to which information remains to be returned. There is no requirement placed on the DHCPv6 server for any particular structure in the OPTION_LQ_COOKIE option.

The conceptual basis of the OPTION_LQ_COOKIE option is that this option is a key which moves through the database and represents the position before which data has been returned and after which data has not been returned. It represents a "scan" through the database for all information to return to a particular LEASEQUERY.

4.5.1. Conceptual Model

The conceptual model for the OPTION_LQ_COOKIE option and bulk queries in general is as follows. Please note that this is a conceptual model only, and does not presuppose a particular implementation approach. Again, this is only a way to think about the OPTION_LQ_COOKIE option, not a recommendation on how to implement it.

Imagine the OPTION_CLIENT_DATA options that the DHCPv6 server would

like to return for a particular LEASEQUERY message. These OPTION_CLIENT_DATA options are ordered in a particular sequence -- a sequence likely known only to the DHCPv6 server. The OPTION_LQ_COOKIE option is essentially a key into this sequence of OPTION_CLIENT_DATA options. Thus, when the DHCPv6 server has more information in the sequence of OPTION_CLIENT_DATA options to return than fits in one message, it returns an OPTION_LQ_COOKIE. The OPTION_LQ_COOKIE is essentially a key to the next OPTION_CLIENT_DATA option that it would return for this query if there were more space in the LEASEQUERY-REPLY message.

When the requestor receives a LEASEQUERY-REPLY containing a OPTION_LQ_COOKIE option, it knows that additional data is available for the query in the original LEASEQUERY message. It processes the data that is received in the LEASEQUERY-REPLY message and then, in order to receive the next message-worth of data, it sends another LEASEQUERY message which MUST contain exactly the same query parameters as the previous message and which MUST contain the OPTION_LQ_COOKIE option and the OPTION_SERVERID option from the last LEASEQUERY-REPLY.

The DHCPv6 server then receives the new LEASEQUERY message and looks for an OPTION_LQ_COOKIE option which it finds. From the key in the OPTION_LQ_COOKIE option it determines where in the sequence of OPTION_CLIENT_DATA options it stopped when filling up the previous LEASEQUERY-REPLY message, and it fills up another LEASEQUERY-REPLY message with the next OPTION_CLIENT_DATA options that fit, and returns an OPTION_LQ_COOKIE option which points to the first OPTION_CLIENT_DATA option that didn't fit into the LEASEQUERY-REPLY message.

This is only a conceptual model, because the DHCPv6 server is not required to keep a list of all of the OPTION_CLIENT_DATA options that it would like to return in response to a particular LEASEQUERY message - that would require a potentially large amount of state to be held for every active sequence of LEASEQUERY query series. A DHCPv6 server implementation may hold such state, but there is no requirement that it do so. The issues that may arise from not holding such state for active queries are discussed below.

4.5.2. Requestor Processing

Whenever a requestor sends a LEASEQUERY message to a DHCPv6 server which contains an OPTION_LQ_COOKIE option, the data within that option MUST be identical to bytes received in an OPTION_LQ_COOKIE option in the last LEASEQUERY-REPLY from that same DHCPv6 server.

The requestor MUST be prepared to process a LEASEQUERY-REPLY message

which contains an OPTION_LQ_COOKIE option, indicating additional data is available. The requestor MUST also be prepared to receive an error when sending any subsequent LEASEQUERY message containing an OPTION_LQ_COOKIE option.

It is possible that some implementations of the LEASEQUERY message will require state memory on the DHCPv6 server, and so a server may encode some state identifier in the OPTION_LQ_COOKIE option. If that state information is not available when the DHCPv6 server receives a LEASEQUERY message with an OPTION_LQ_COOKIE option, then the DHCPv6 server MUST return a StaleCookie error status. This is only an example of why a DHCPv6 server might return an error to a LEASEQUERY message which contains an OPTION_LQ_COOKIE option -- it is not meant to restrict the DHCPv6 server in any way. An error may be returned on any LEASEQUERY message with an OPTION_LQ_COOKIE and the requestor MUST be able to deal with it.

In the event that the requestor who sent a LEASEQUERY receives a StaleCookie error on a LEASEQUERY-REPLY message from the DHCPv6 server, then the requestor SHOULD restart the LEASEQUERY exchange message from the beginning. In particular, the requestor MUST NOT include an OPTION_LQ_COOKIE option in the subsequent LEASEQUERY message. It SHOULD NOT throw away all of the existing information already received, but it MUST be prepared to receive different and possibly more up to date information than it received previously.

The requestor needs to recognize that information that is returned in multiple LEASEQUERY-REPLYs resulting from a single query MAY not be from one instant in time, but rather MAY represent a series of snapshots of different segments of the database.

The requestor SHOULD use the OPTION_SERVER_RSN and OPTION_CLT_TIME options to distinguish more current from less current OPTION_CLIENT_DATA information as discussed in Section 4.3.4.

In the event that the requestor is sending in a LEASEQUERY message in order to perform an "Anticipatory Leasequery", as described in Section 3.2, presumably there is some internal database it is attempting to reestablish by using the LEASEQUERY messages. In the usual case, this internal database is maintained by a mechanism other than leasequery (e.g. [9]), and the LEASEQUERY messages are just used to reestablish the current state of this internal database. In the cases where this situation exists, the requestor SHOULD reestablish the mechanism used to maintain this internal database prior to the time that it sends the first anticipatory LEASEQUERY message.

If this is done, then there may well be some cases where the data obtained from a LEASEQUERY exchange is different than the data

determined from the database maintenance mechanism; how to handle this is described in Section 4.3.4.

4.5.3. Server Processing

When the DHCPv6 server receives a LEASEQUERY message containing a OPTION_LQ_COOKIE option and an OPTION_SERVERID option, it compares the OPTION_SERVERID option with its DUID to determine if the LEASEQUERY message is directed at itself. It then uses the OPTION_LQ_COOKIE and OPTION_LQ_QUERY options to generate additional OPTION_CLIENT_DATA options to return to the requestor. The DHCPv6 server MUST be able to generate output in such a way that it can use information in the OPTION_LQ_COOKIE option to determine which information it has already returned and which information has yet to be returned as a result of the query embodied in the LEASEQUERY message.

One issue is that the underlying data may well have changed in the time since the first query was received and processed, yielding several possible inconsistencies in the data returned to the requestor.

The most obvious issue is that the information contained in the OPTION_LQ_COOKIE option may be a key of some kind to the next information that was to be returned to the requestor but did not fit in the LEASEQUERY-REPLY message. By the time that the subsequent LEASEQUERY message is received, however, that information may not be supposed to be returned to the requestor. In this case, the DHCPv6 server MUST return the next OPTION_CLIENT_DATA option not already returned to sender of the LEASEQUERY message that is beyond the "key" in the OPTION_LQ_COOKIE.

In situations where ambiguity exists, the DHCPv6 server MAY return duplicate information but SHOULD NOT make this a common occurrence -- or it will be possible for a query series to never terminate. The DHCPv6 server MUST ensure that a query series will terminate and not continue indefinitely.

The DHCPv6 server processing a LEASEQUERY message which contains an OPTION_LQ_COOKIE MUST NOT return an error if the only problem with the data in the OPTION_LQ_COOKIE option is that it refers to a particular chunk of client data that is not now supposed to be returned as a response to this LEASEQUERY. In this case, the DHCPv6 server MUST return the next available chunk of client data that should now be returned from this LEASEQUERY. If these guidelines are not followed, it would be possible to end up with a situation where a series of LEASQUERY with OPTION_LQ_COOKIE option exchanges might never terminate.

If there is OPTION_CLIENT_DATA option information that was not sent in the initial response to the LEASEQUERY message but which is now available, but precedes the "key" in the OPTION_LQ_COOKIE option, the DHCPv6 server SHOULD NOT return it. Likewise if there is information that was returned in the initial response to the LEASEQUERY message but which would not now be returned as a response from the identical LEASEQUERY message, the DHCPv6 server SHOULD NOT do anything about this.

5. Security Considerations

The senders of LEASEQUERY messages are expected to be within the same security domain as the DHCPv6 server. As such, the security threat to DHCPv6 leasequery is inherently an insider threat. However, this document doesn't prohibit entities in external security domains from sending LEASEQUERY messages to DHCPv6 servers. Regardless of the network configuration, however, the potential attacks by insiders and outsiders are the same.

If the requestor is an access concentrator, DHCPv6 leasequery security SHOULD follow security between the relay agent and the DHCPv6 server as described in [2] Sections 21.1 and 22.11. Requestors are essentially a DHCPv6 client for the purposes of the LEASEQUERY message. Thus, DHCPv6 authentication [2] is also an appropriate mechanism for securing LEASEQUERY and LEASEQUERY-REPLY messages.

Access concentrators are expected to be common leasequery requestors. Access concentrators that use DHCPv6 gleaning (i.e., [9]), refreshed with LEASEQUERY messages, will maintain accurate client/binding information. This ensures that the access concentrator can forward data traffic to the intended destination in the broadband access network, can perform IPv6 source address verification of datagrams from the access network, and can encrypt traffic that can only be decrypted by the intended access modem (e.g., [BPI] and [BPI+]). Thus, the LEASEQUERY message allows an access concentrator to provide considerably enhanced security. DHCPv6 servers SHOULD prevent exposure of their information (particularly the mapping of hardware address to IPv6 address, which can be an invasion of broadband subscriber privacy) by employing the techniques detailed in [2], Section 21, "Authentication of DHCP Messages".

DHCPv6 servers SHOULD also provide for the ability to restrict the information that they make via leasequery, as described in Section 4.4.3.

DHCPv6 servers supporting LEASEQUERY SHOULD ensure that they cannot

be successfully attacked by being flooded with large quantities of LEASEQUERY messages in a short time. In some environments, it may be appropriate to configure a DHCPv6 server with the IPv6 source addresses of the relay agents for which it may respond to LEASEQUERY messages, thereby allowing it to respond only to requests from only a handful of relay agents. This does not provide any true security, but may be useful to thwart unsophisticated attacks of various sorts.

Replayed messages can represent a DOS attack through exhaustion of processing resources, bogus leasequery requestors can send a lot of LEASEQUERY messages to overwhelm a DHCPv6 server, thus preventing the server from serving legitimate and regular DHCPv6 clients as well as legitimate DHCPv6 leasequery requestors, denying configurations to legitimate DHCPv6 clients as well lease information to legitimate DHCPv6 leasequery requestors.

One attack specific to an access concentrator as a requestor is the establishment of a malicious server with the intent of providing incorrect lease or route information to the access concentrator, thwarting source IPv6 address verification and preventing correct routing.

The use of the OPTION_SERVER_RSN option does provide an attacker that also knows the server's DUID the ability to effectively lock out future updates from the real server by supply a large sequence number.

6. IANA Considerations

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

- LEASEQUERY
- LEASEQUERY-REPLY

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

- OPTION_LQ_QUERY
- OPTION_REPLY_SIZE
- OPTION_LQ_COOKIE
- OPTION_CLIENT_DATA
- OPTION_CLT_TIME

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

```
   UnknownQueryType
   MalformedQuery
   StaleCookie
   TooShort
   NotConfigured
   NotAllowed
```

IANA is requested to create a new registry for the OPTION_LQ_QUERY option query-type codes in the registry maintained in http://www.iana.org/assignments/dhcpv6-parameters with the following initial assignments:

```
   QUERY_BY_CLIENTID      1
   QUERY_BY_ADDRESS       2
   QUERY_BY_PREFIX        3
   QUERY_BY_LINK          4
   QUERY_BY_REMOTE_ID     5
   QUERY_BY_DEVICE_ID     6
```

7. Acknowledgements

Thanks to Ralph Droms, Richard Johnson, Josh Littlefield, Hemant Singh, and Pak Siripunkaw for their input, ideas, and review during the production of this document.

8. References

8.1. Normative References

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

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

[3] Woundy, R. and K. Kinnear, "Dynamic Host Configuration Protocol (DHCP) Leasequery", RFC 4388, February 2006.

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

8.2. Informative References

[5] Droms, R., "Dynamic Host Configuration Protocol", RFC 2131, March 1997.

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

[7] Volz, B., "DHCPv6 Relay Agent Remote ID Option (draft-ietf-dhc-dhcpv6-remoteid-*)", March 2006.

[8] Droms, R., "DHCPv6 Device ID Option (draft-droms-dhc-dhcpv6-deviceid-*)", June 2006.

[9] Droms, R., Volz, B., and O. Troan, "DHCP Relay Agent Assignment Notification Option (draft-ietf-dhc-dhcpv6-agentopt-delegate-*)", June 2006.

Authors' Addresses

John Jason Brzozowski
Comcast Cable
1800 Bishops Gate Boulevard
Mt. Laurel, NJ 08054
USA

Phone: +1 856 324-2671
Email: john_brzozowski@cable.comcast.com

Kim Kinnear
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

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

Bernard Volz
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

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

Shengyou Zeng
Cisco Systems, Inc.
1414 Massachusetts Ave.
Boxborough, MA 01719
USA

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

Intellectual Property Statement

The IETF takes no position regarding the validity or scope of any Intellectual Property Rights or other rights that might be claimed to pertain to the implementation or use of the technology described in this document or the extent to which any license under such rights might or might not be available; nor does it represent that it has made any independent effort to identify any such rights. Information on the procedures with respect to rights in RFC documents can be found in BCP 78 and BCP 79.

Copies of IPR disclosures made to the IETF Secretariat and any assurances of licenses to be made available, or the result of an attempt made to obtain a general license or permission for the use of such proprietary rights by implementers or users of this 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

This document and the information contained herein are provided on an "AS IS" basis and THE CONTRIBUTOR, THE ORGANIZATION HE/SHE REPRESENTS OR IS SPONSORED BY (IF ANY), THE INTERNET SOCIETY AND THE INTERNET ENGINEERING TASK FORCE DISCLAIM ALL WARRANTIES, EXPRESS OR IMPLIED, INCLUDING BUT NOT LIMITED TO ANY WARRANTY THAT THE USE OF THE INFORMATION HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Copyright Statement

Copyright (C) The Internet Society (2006). This document is subject to the rights, licenses and restrictions contained in BCP 78, and except as set forth therein, the authors retain all their rights.

Acknowledgment

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