Network Working Group
Requests for Comments: 2730
Category: Standards Track

S. Hanna
Sun Microsystems, Inc.
B. Patel
Intel Corp.
M. Shah
Microsoft Corp.

December 1999

Multicast Address Dynamic Client Allocation Protocol (MADCAP)

Status of this Memo

Copyright Notice

Abstract

This document defines a protocol, Multicast Address Dynamic Client Allocation Protocol (MADCAP), that allows hosts to request multicast addresses from multicast address allocation servers.

1. Introduction

Multicast Address Dynamic Client Allocation Protocol (MADCAP) is a protocol that allows hosts to request multicast address allocation services from multicast address allocation servers. This protocol is part of the Multicast Address Allocation Architecture being defined by the IETF Multicast Address Allocation Working Group. However, it may be used separately from the rest of that architecture as appropriate.

1.1. 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 RFC 2119 [9].

Constants used by this protocol are shown as [NAME-OF-CONSTANT], and summarized in Appendix B.

1.2. Definitions

This specification uses a number of terms that may not be familiar to the reader. This section defines some of these and refers to other documents for definitions of others.

MADCAP client or client
A host requesting multicast address allocation services via MADCAP.

MADCAP server or server
A host providing multicast address allocation services via MADCAP.

Multicast
IP Multicast, as defined in [11] and modified in [12].

Multicast Address
An IP multicast address or group address, as defined in [11] and [13]. An identifier for a group of nodes.

Multicast Scope
A range of multicast addresses configured so that traffic sent to these addresses is limited to some subset of the internetwork. See [3] and [13].

Scope ID
The lowest numbered address in a multicast scope. This definition applies only within this document.

Scope Zone
One multicast scope may have several instances, which are known as Scope Zones or zones, for short.

For instance, an organization may have multiple sites. Each site might have its own site-local Scope Zone, each of which would be an instance of the site-local Scope. However, a given interface on a given host would only ever be in at most one instance of a given scope. Messages sent by a host in a site-local Scope Zones to an address in the site-local Scope would be limited to the site-local Scope Zone containing the host.

Zone Name
A human readable name for a Scope Zone. An ISO 10646 character string with an RFC 1766 [6] language tag. One zone may have several zone names, each in a different language. For instance, a zone for use within IBM's locations in Switzerland might have the names "IBM Suisse", "IBM Switzerland", "IBM Schweiz", and "IBM Svizzera" with language tags "fr", "en", "de", and "it".

Multicast Scope List
A list of multicast scope zones.

Since it can be difficult to determine which multicast scope zones are in effect, MADCAP clients can ask MADCAP servers to supply a Multicast Scope List listing all of the zones available to the client. For each scope zone, the list includes the range of multicast addresses for this scope, a maximum TTL or hop count to be used for this scope, and one or more zone names for this scope zone.

This definition applies only within this document.

1.3. Motivation and Protocol Requirements

For multicast applications to be deployed everywhere, there is a need to define a protocol that any host may use to allocate multicast addresses. Here are the requirements for such a protocol.

Quick response: The host should be able to allocate a multicast address and begin to use it promptly.

Low network load: Hosts that are not allocating or deallocating multicast addresses at the present time should not need to send or receive any network traffic.

Support for intermittently connected or power managed systems: Hosts should be able to be disconnected from the network, powered off, or otherwise inaccessible except during the brief period during which they are allocating a multicast address.

Multicast address scopes: The protocol must be able to allocate both the administratively scoped and globally scoped multicast addresses.

Efficient use of address space: The multicast address space is fairly small. The protocol should make efficient use of this scarce resource.

Authentication: Because multicast addresses are scarce, it is important to protect against hoarding of these addresses. One way to do this is by authenticating clients. This is also a key prerequisite for establishing policies.

Policy neutral: Allocation policies (such as who can allocate addresses) should not be dictated by the protocol.

Conferencing support: When allocating an address for use in a conferencing environment, members of the conference should be able to modify a multicast address lease used for the conference.

1.4. Relationship with DHCP

MADCAP was originally based on DHCP. There are still some similarities and it may be possible to share some code between a DHCP implementation and a MADCAP implementation. However, MADCAP is completely separate from DHCP, with no dependencies between the two and many significant differences.

1.5. Protocol Overview

MADCAP is built on a client-server model, where hosts request address allocation services from address allocation servers. When a MADCAP client wishes to request a service, it unicasts or multicasts a message to one or more MADCAP servers, each of which optionally responds with a message unicast to the client.

All messages are UDP datagrams. The UDP data contains a fixed length header and a variable length options field. Options are encoded in a type-length-value format with two octets type and value fields. The fixed fields are version, msgtype (message type), addrfamily (address family), and xid (transaction identifier).

Retransmission is handled by the client. If a client sends a message and does not receive a response, it may retransmit its request a few times using an exponential backoff. To avoid executing the same client request twice when a retransmitted request is received, servers cache responses for a short period of time and resend cached responses upon receiving retransmitted requests.

Each request contains a msgtype, an xid, and a Lease Identifier option. Clients must ensure that this triple is probably unique across all MADCAP messages received by a MADCAP server over a period of [XID-REUSE-INTERVAL] (10 minutes). This allows the MADCAP server to use this triple as the key in its response cache.

Messages sent by servers include the xid included in the original request so that clients can match up responses with requests.

The msgtype field is a single octet that defines the "type" of a MADCAP message. Currently defined message types are listed in Table 2. They are: DISCOVER, OFFER, REQUEST, RENEW, ACK, NAK, RELEASE, and

GETINFO. DISCOVER, REQUEST, RENEW, RELEASE, and GETINFO messages are only sent by a client. OFFER, ACK, and NAK messages are only sent by a server.

The REQUEST, RENEW, and RELEASE messages are used to request, renew, or release a lease on one or more multicast addresses. A client unicasts one of these messages to a server and the server responds with an ACK or a NAK.

The GETINFO message is used to request information, such as the multicast scope list, or to find MADCAP servers. A client may unicast an GETINFO message to a MADCAP server. However, it may not know the IP address of any MADCAP server. In that case, it will multicast an GETINFO message to a MADCAP Server Multicast Address and all servers that wish to respond will send a unicast ACK or NAK back to the client.

Each multicast scope has an associated MADCAP Server Multicast Address. This address has been reserved by the IANA as the address with a relative offset of -1 from the last address of a multicast scope. MADCAP clients use this address to find MADCAP servers.

The DISCOVER message is a message used to discover MADCAP servers that can probably satisfy a REQUEST. DISCOVER messages are always multicast. Servers that can probably satisfy a REQUEST corresponding to the parameters supplied in the DISCOVER message temporarily reserve the addresses needed and send a unicast OFFER back to the client. The client selects a server with which to continue and sends a multicast REQUEST including the server's Server Identifier to the same multicast address used for the DISCOVER. The chosen server responds with an ACK or NAK and the other servers stop reserving the addresses they were temporarily holding.

For detailed descriptions of typical protocol exchanges, consult Appendix A.

MADCAP is a mechanism rather than a policy. MADCAP allows local system administrators to exercise control over configuration parameters where desired. For example, MADCAP servers may be configured to limit the number of multicast addresses allocated to a single client. Properly enforcing such a limit requires cryptographic security, as described in the Security Consideration section.

MADCAP requests from a single host may be sent on behalf of different applications with different needs and requirements. MADCAP servers MUST NOT assume that because one request from a MADCAP client supports a particular optional feature (like Retry After), future requests from that client will also support that optional feature.

2. Protocol Description

The MADCAP protocol is a client-server protocol. In general, the client unicasts or multicasts a message to one or more servers, which optionally respond with messages unicast to the client.

A reserved port number dedicated for MADCAP is used on the server (port number 2535, as assigned by IANA). Any port number may be used on client machines. When a MADCAP server sends a message to a MADCAP client, it MUST use a destination port number that matches the source port number provided by the client in the message that caused the server to send its message.

The next few sections describe the MADCAP message format and message types. A full list of MADCAP options is provided in section 3.

2.1. Message Format

Figure 1 gives the format of a MADCAP message and Table 1 describes each of the fields in the MADCAP message. The numbers in parentheses indicate the size of each field in octets. The names for the fields given in the figure will be used throughout this document to refer to the fields in MADCAP messages.

All multi-octet quantities are in network byte-order.

Any message whose UDP data is too short to hold this format (at least 12 bytes) MUST be ignored.

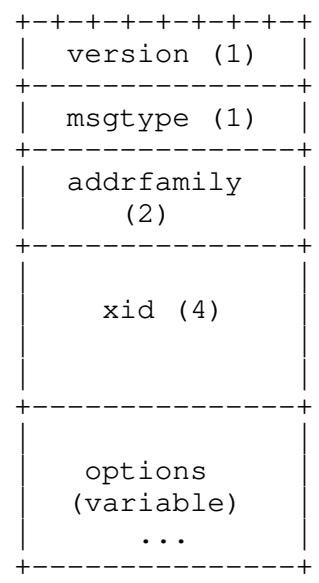


Figure 1: Format of a MADCAP message

FIELD	OCTETS	DESCRIPTION
version	1	Protocol version number (zero for this specification)
msgtype	1	Message type (DISCOVER, GETINFO, etc.)
addrfamily	2	Address family (IPv4, IPv6, etc.)
xid	4	Transaction ID
options	var	Options field

Table 1: Description of fields in a MADCAP message

2.1.1. The version field

The version field must always be zero for this version of the protocol. Any messages that include other values in this field MUST be ignored.

2.1.2. The msgtype field

The msgtype field defines the "type" of the MADCAP message.

For more information about this field, see section 2.2.

2.1.3. The addrfamily field

The addrfamily field defines the default address family (such as IPv4 or IPv6) for this MADCAP message, using the address family numbers defined in by the IANA (including those defined in [10]). Unless otherwise specified, all addresses included in the message will be from this family.

2.1.4. The xid field

The xid field is a transaction identifier. This number MUST be chosen by the client so that the combination of xid, msgtype, and Lease Identifier is unique across all MADCAP messages received by a MADCAP server over a period of [XID-REUSE-INTERVAL] (10 minutes).

The xid field is used by the client and server to associate messages and responses between a client and a server. Before a client sends a message, it chooses a number to use as an xid. The technique used to choose an xid is implementation-dependent, but whatever technique is used MUST make it unlikely that the same combination of xid, msgtype, and Lease Identifier will be used for two different messages within [XID-REUSE-INTERVAL] (even across multiple clients which do not communicate among themselves). This allows enough time for the message to be dropped from all server response caches (as described in the next few paragraphs) and for any network delays to be accomodated.

The RECOMMENDED technique for choosing an xid is to choose a random four octet number as the first xid in a session and increment this value each time a new xid is needed. The random number chosen need not be cryptographically random. The random number may be chosen via any suitable technique, such as the one described in section A.6 of RFC 1889 [14].

When a server responds to a client message, it MUST use the same xid value in the response that the client used in the request. This allows the client to associate responses with the message that they are responding to.

When retransmitting messages (as described in section 2.3), the client MUST retransmit them without changing them, thereby using the same xid and and Lease Identifier.

If a server receives a message with the same xid, msgtype, and Lease Identifier as one received within [RESPONSE-CACHE-INTERVAL], it MUST treat this message as a retransmission of the previously received one and retransmit the response, if any. After [RESPONSE-CACHE-INTERVAL], the server may forget about the previously received message and treat

any retransmissions of this message as if they were new messages. Of course, a server need not cache a message if it ends up ignoring that message. However, performance gains may be achieved by doing so.

This avoids retransmissions causing multiple allocations, since requests are not idempotent. An appropriate value for [RESPONSE-CACHE-INTERVAL] would be sixty seconds, but it may have any value from zero seconds to 300 seconds (five minutes) and may be adjusted dynamically according to resource constraints on the server. However, using a value less than sixty seconds is NOT RECOMMENDED because this is the normal client retransmission period.

2.1.5. The options field

The options field consists of a list of tagged parameters that are called "options". All options consist of a two octet option code and a two octet option length, followed by the number of octets specified by the option length. In the case of some options, the length field is a constant but must still be specified.

The option field MUST contain a sequence of options with the last one being the End option (option code 0). Any message whose options field does not conform to this syntax MUST be ignored.

Any MADCAP client or server sending a MADCAP message MAY include any of the options listed in section 3, subject to the restrictions in Table 5 and elsewhere in this document. They MAY also include other MADCAP options that are defined in the future. A MADCAP client or server MUST NOT include more than one option with the same option type in one MADCAP message.

All MADCAP clients and servers MUST recognize all options listed in this document and behave in accordance with this document when receiving and processing any of these options. Any unrecognized options MUST be ignored and the rest of the message processed as if the unknown options were not present. If a MADCAP server receives a message that does not conform to the requirements of this document (for instance, not including all required options), an Invalid Request error MUST be generated and processed in the manner described in section 2.6. If a MADCAP client receives a message that does not conform to the requirements of this document, it MUST ignore the message.

The order of options within a message has no significance and any order MUST be supported in an equivalent manner, with the exception that the End option must occur once per message, as the last option in the option field.

New MADCAP option codes may only be defined by IETF Consensus, as described in section 5.

2.2. Message Types

The msgtype field defines the "type" of a MADCAP message. Legal values for this field are:

Value	Message Type
1	DISCOVER
2	OFFER
3	REQUEST
4	RENEW
5	ACK
6	NAK
7	RELEASE
8	GETINFO

Table 2: MADCAP message types

Throughout this document, MADCAP messages will be referred to by the type of the message; e.g., a MADCAP message with a message type of 8 will be referred to as an GETINFO message.

Here are descriptions of the MADCAP message types. Table 5, which appears at the beginning of section 3, summarizes which options are allowed with each message type.

MADCAP clients and servers MUST handle all MADCAP message types defined in this document in a manner consistent with this document.

If a MADCAP server receives a message whose message type it does not recognize, an Invalid Request error MUST be generated and processed in the manner described in section 2.6. If a MADCAP client receives a message whose message type it does not recognize, it MUST ignore the message.

Note, however, that under some circumstances this document requires or suggests that clients or servers ignore messages with certain message types even though they may be recognized. For instance, clients that do not send DISCOVER messages SHOULD ignore OFFER messages. Also, secure servers SHOULD ignore DISCOVER messages and all servers SHOULD ignore DISCOVER messages that they cannot satisfy.

New MADCAP message types may only be defined by IETF Consensus, as described in section 5.

2.2.1. GETINFO

The GETINFO message is used by a MADCAP client that wants to acquire configuration parameters, especially a multicast scope list. This message also allows a client to determine which servers are likely to be able to handle future requests.

The MADCAP client sends out an GETINFO message. The message may be unicast to a particular MADCAP server or multicast to a MADCAP Server Multicast Address. For more details about the MADCAP Server Multicast Address, see section 2.10.

If a server receives an GETINFO message and it can process the request successfully, it MUST unicast an ACK message to the client. All GETINFO messages MUST include an Option Request List option. The server SHOULD try to include the specified options in its response, but is not required to do so (especially if it does not recognize them).

If a server receives an GETINFO message and it does not process the request successfully, it MUST generate and process an error in the manner described in section 2.6.

If a client sends an GETINFO message and does not receive any ACK messages in response, it SHOULD resend its GETINFO message, as described in section 2.3.

When a MADCAP client sends an GETINFO message, it MAY include the Requested Language option, which specifies which language the client would prefer for the zone names in the Multicast Scope List. The proper way to handle this tag with respect to zone names is discussed in the definition of the Multicast Scope List option.

2.2.2. DISCOVER

The DISCOVER message is a multicast message sent by a MADCAP client that wants to discover MADCAP servers that can probably satisfy a REQUEST.

MADCAP clients are not required to use the DISCOVER message. They MAY employ other methods to find MADCAP servers, such as sending a multicast GETINFO message, caching an IP address that worked in the past or being configured with an IP address. Using the DISCOVER message has the particular advantage that it allows clients to receive responses from all servers that can satisfy the request.

The MADCAP client begins by sending a multicast DISCOVER message to a MADCAP Server Multicast Address. Any servers that wish to assist the client respond by sending a unicast OFFER message to the client. If a server can only process the request with a shorter lease time or later start time than the client requested, it SHOULD send an OFFER message with the lease time or start time that it can offer. However, it MUST NOT offer a lease time shorter than the minimum lease time specified by the client or a start time later than the maximum start time specified by the client.

For more details about the MADCAP Server Multicast Address, see section 2.10.

If a client sends a DISCOVER message and does not receive any OFFER messages in response, the client SHOULD retransmit its DISCOVER message, as described in section 2.3.

If a client sends a DISCOVER message and receives one or more OFFER messages in response, it SHOULD select the server it wants to use (if any) and send a multicast REQUEST message identifying that server within [DISCOVER-DELAY] after receiving the first OFFER message. See section 2.2.4 for more information about the REQUEST message.

The mechanism used by the client in selecting the server it wants to use is implementation dependent. The client MAY choose the first acceptable response or it MAY wait some period of time (no more than [DISCOVER-DELAY]) and choose the best response received in that period of time (if the first response has a smaller lease time than requested, for instance).

The value of [DISCOVER-DELAY] is also implementation dependent, but the RECOMMENDED value is the current retransmit timer, as specified in section 2.3. Waiting too long (approaching [OFFER-HOLD]) may cause servers to drop the addresses they have reserved.

When a MADCAP client sends a DISCOVER message, it MAY include the Lease Time, Minimum Lease Time, Start Time, Maximum Start Time, Number of Addresses Requested, and List of Address Ranges options, describing the addresses it wants to receive. However, it need not include any of these options. If one of these options is not included, the server will provide the appropriate default (maximum available for Lease Time, no minimum for Minimum Lease Time, as soon as possible for Start Time, no maximum for Maximum Start Time, one for Number of Addresses Requested, and any addresses available for List of Address Ranges). The Multicast Scope option MUST be included in the DISCOVER message so that the server knows what scope should be used. The Current Time option MUST be included if the Start Time or Maximum Start Time options are included. The Lease Identifier option

MUST always be included.

2.2.3. OFFER

The OFFER message is a unicast message sent by a MADCAP server in response to a DISCOVER message that it can probably satisfy.

A MADCAP server is never required to send an OFFER message in response to a DISCOVER message. For instance, it may not be able to satisfy the client's request or it may have been configured to respond only to certain types of DISCOVER messages or not to respond to DISCOVER messages at all.

If a MADCAP server decides to send an OFFER message, it MUST include the Lease Time and Multicast Scope options, describing the addresses it is willing to provide. However, it need not include the List of Address Ranges option. If the List of Address Ranges Allocated option is not included, it is assumed that the server is willing to provide the number of addresses that the client requested. If the Start Time option is not included, it is assumed that the server is willing to provide the start time requested by the client (if any). The Current Time option MUST be included if the Start Time option is included.

If a server can process the request with a shorter lease time or later start time than the client requested, it SHOULD send an OFFER message with the lease time or start time that it can offer. However, it MUST NOT offer a lease time shorter than the minimum lease time specified by the client or a start time later than the maximum start time specified by the client.

If the server sends an OFFER message, it SHOULD attempt to hold enough addresses to complete the transaction. If it receives a multicast REQUEST message with the same Lease Identifier option as the DISCOVER message for which it is holding these addresses and a Server Identifier option that does not match its own, it SHOULD stop holding the addresses. The server SHOULD also stop holding the addresses after an appropriate delay [OFFER-HOLD] if the transaction is not completed. The value of this delay is implementation-specific, but a value of at least 60 seconds is RECOMMENDED.

As with all messages sent by the server, the xid field MUST match the xid field included in the client request to which this message is responding. The Lease Identifier option MUST be included, with the value matching the one included in the client request. The Server Identifier option MUST be included, with the value being the server's IP address. And the packet MUST NOT be retransmitted.

2.2.4. REQUEST

The REQUEST message is used by a MADCAP client that wants to allocate one or more multicast addresses. It is not used for renewing an existing lease. The RENEW message is used for that.

If a REQUEST message is completing a transaction initiated by a DISCOVER message, the following procedure MUST be followed so that all MADCAP servers know which server was selected. The client MUST multicast a REQUEST message to the same MADCAP Server Multicast Address that the DISCOVER message was sent to. The same Lease Identifier used in the DISCOVER message MUST be used in the REQUEST message. Also, the Server Identifier option MUST be included, using the Server Identifier of the server selected.

If a REQUEST message is not completing a transaction initiated by a DISCOVER message, the REQUEST message MUST be unicast to the MADCAP server that the client wants to use. In this case, the Server Identifier option MAY be included, but need not be.

If the selected server can process the request successfully, it SHOULD unicast an ACK message to the client. Otherwise, it SHOULD generate and process an error in the manner described in section 2.6. If a server can process the request with a shorter lease time or later start time than the client requested, it SHOULD send an ACK message with the lease time or start time that it can offer. However, it MUST NOT offer a lease time shorter than the minimum lease time specified by the client or a start time later than the maximum start time specified by the client.

When a MADCAP client sends a REQUEST message, it MAY include the Lease Time, Minimum Lease Time, Start Time, Maximum Start Time, Number of Addresses Requested, and List of Address Ranges options, describing the addresses it wants to receive. However, it need not include any of these options. If one of these options is not included, the server will provide the appropriate default (maximum available for Lease Time, no minimum for Minimum Lease Time, as soon as possible for Start Time, no maximum for Maximum Start Time, one for Number of Addresses Requested, and any addresses available for List of Address Ranges). The Multicast Scope option MUST be included in the REQUEST message so that the server knows what scope should be used. The Current Time option MUST be included if the Start Time or Maximum Start Time options are included.

If a client sends a REQUEST message and does not receive any ACK or NAK messages in response, the client SHOULD resend its REQUEST message, as described in section 2.3.

If the server responds with a NAK or fails to respond within a reasonable (implementation-dependent) delay [NO-RESPONSE-DELAY], the client MAY try to find another server by sending a DISCOVER message with another xid or sending a REQUEST message with another xid to another server. The RECOMMENDED value for [NO-RESPONSE-DELAY] is 60 seconds.

2.2.5. ACK

The ACK message is used by a MADCAP server to respond affirmatively to an GETINFO, REQUEST, or RELEASE message. The server unicasts the ACK message to the client from which it received the message to which it is responding.

The set of options included with an ACK message differs, depending on what sort of message it is responding to.

If the ACK message is responding to an GETINFO message, it SHOULD include any options requested by the client using the Option Request List option.

If the ACK message is responding to a REQUEST message, it MUST include Lease Time, Multicast Scope, and List of Address Ranges options. It MAY include a Start Time option. If a Start Time option is included, a Current Time option MUST also be included. If no Start Time option is included, the lease is assumed to start immediately.

If the ACK message is responding to a RENEW message, it MUST include Lease Time, Multicast Scope, and List of Address Ranges options. It MAY include a Start Time option. If a Start Time option is included, a Current Time option MUST also be included. If no Start Time option is included, the lease is assumed to start immediately.

If the ACK message is responding to a RELEASE message, it MUST only include Server Identifier and Lease Identifier options.

As with all messages sent by the server, the xid field MUST match the xid field included in the client request to which this message is responding. The Lease Identifier option MUST be included, with the value matching the one included in the client request. The Server Identifier option MUST be included, with the value being the server's IP address. And the packet MUST NOT be retransmitted.

2.2.6. NAK

The NAK message is used by a MADCAP server to respond negatively to a message. The server unicasts the NAK message to the client from which it received the message to which it is responding.

As with all messages sent by the server, the xid field MUST match the xid field included in the client request to which this message is responding. The Lease Identifier option MUST be included, with the value matching the one included in the client request. The Server Identifier option MUST be included, with the value being the server's IP address. The Error option MUST be included with an error code indicating what went wrong. And the packet MUST NOT be retransmitted.

2.2.7. RENEW

The RENEW message is used by a MADCAP client that wants to renew a multicast address lease, changing the lease time or start time.

The client unicasts the RENEW message to a MADCAP server. If the server can process the request successfully, it SHOULD unicast an ACK message to the client. Otherwise, it MUST generate and process an error in the manner described in section 2.6.

The lease to be renewed is whichever one was allocated with a Lease Identifier option matching the one provided in the RENEW message.

When a MADCAP client sends a RENEW message, it MAY include the Lease Time, Minimum Lease Time, Start Time, and Maximum Start Time options, describing the new lease it wants to receive. However, it need not include any of these options. If one of these options is not included, the server will provide the appropriate default (maximum available for Lease Time, no minimum for Minimum Lease Time, as soon as possible for Start Time, and no maximum for Maximum Start Time). The Current Time option MUST be included if the Start Time or Maximum Start Time options are included.

If a client sends a RENEW message and does not receive any ACK or NAK messages in response, the client SHOULD resend its RENEW message, as described in section 2.3.

If the server responds with a NAK or fails to respond within a reasonable (implementation-dependent) delay [NO-RESPONSE-DELAY], the client MAY send a RENEW message with another xid to another server, provided that the Server Mobility feature was used in the original REQUEST message and that this feature is required for the subsequent RENEW message sent to another server. For more information about the Server Mobility feature, see section 2.13.1. The RECOMMENDED value for [NO-RESPONSE-DELAY] is 60 seconds.

2.2.8. RELEASE

The RELEASE message is used by a MADCAP client that wants to deallocate one or more multicast addresses before their lease expires.

The client unicasts the RELEASE message to the MADCAP server from which it allocated the addresses. If the selected server can process the request successfully, it MUST unicast an ACK message to the client. Otherwise, it MUST generate and process an error in the manner described in section 2.6.

The lease to be released is whichever one was allocated with a Lease Identifier option matching the one provided in the RELEASE message. It is not possible to release only part of the addresses in a single lease.

If a client sends a RELEASE message and does not receive any ACK or NAK messages in response, the client SHOULD resend its RELEASE message, as described in section 2.3.

If the server responds with a NAK or fails to respond within a reasonable (implementation-dependent) delay [NO-RESPONSE-DELAY], the client MAY send a RELEASE message with another xid to another server, provided that the Server Mobility feature was used in the original REQUEST message and that this feature is required for the subsequent RELEASE message sent to another server. For more information about the Server Mobility feature, see section 2.13.1. The RECOMMENDED value for [NO-RESPONSE-DELAY] is 60 seconds.

2.3. Retransmission

MADCAP clients are responsible for all message retransmission. The client MUST adopt a retransmission strategy that incorporates an exponential backoff algorithm to determine the delay between retransmissions. The delay between retransmissions SHOULD be chosen to allow sufficient time for replies from the server to be delivered based on the characteristics of the internetwork between the client and the server.

The RECOMMENDED algorithm is to use a 4 second delay before the first retransmission and to double this delay for each successive retransmission, with a maximum delay of 16 seconds and a maximum of three retransmissions. If an initial transmission was sent at time (in seconds) t and no responses were received, subsequent transmissions would be at t+4, t+12, and t+28. If no response has been received by t+60, the client would stop retransmitting and take another course of action (such as logging an error or sending a

message to another address.

The client MAY provide an indication of retransmission attempts to the user as an indication of the progress of the process. The client MAY halt retransmission at any point.

2.4. The Lease Identifier

The Lease Identifier option is included in each MADCAP message. Its value is used to identify a lease and MUST be unique across all leases requested by all clients in a multicast address allocation domain.

The first octet of the Lease Identifier is the Lease Identifier type. Table 3 lists the Lease Identifier types defined at this time and sections 2.4.1 and 2.4.2 describe these Lease Identifier types.

New MADCAP Lease Identifier types may only be defined by IETF Consensus, as described in section 5.

Lease Identifier Type	Name
0	Random Lease Identifier
1	Address-Specific Lease Identifier

Table 3: MADCAP Lease Identifier Types

The MADCAP server does not need to parse the Lease Identifier. It SHOULD use the Lease Identifier only as an opaque identifier, which must be unique for each lease. The purpose of defining different Lease Identifier types is to allow MADCAP clients that already have a globally unique address to avoid the possibility of Lease Identifier collisions by using this address together with a client-specific identifier. MADCAP clients that do not have a globally unique address SHOULD use Lease Identifier type 0.

In addition to associating client and server messages (along with the msgtype and xid fields, as described in the next section), the Lease Identifier is used to determine which lease a RENEW or RELEASE request refers to. MADCAP servers SHOULD match the Lease Identifier included in a RENEW or RELEASE message with the Lease Identifier used in an initial REQUEST message. If the Lease Identifier does not match, a MADCAP server MUST generate and process a Lease Identifier Not Recognized error in the manner described in section 2.6.

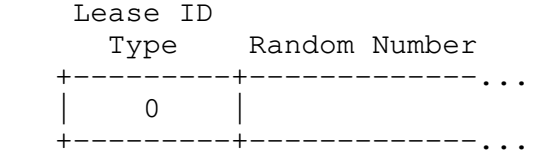
For conferencing applications, it may be desirable to allow conference participants to modify a lease used for the conference. The Shared Lease Identifier feature code is used to support this requirement. If this feature code was requested by the client and implemented by the server when the lease was allocated, the server SHOULD disable any authentication requirements and allow any client that knows the Lease Identifier to modify the lease.

As described in the Security Considerations section, MADCAP security is not terribly useful without admission control in the multicast routing infrastructure. However, if MADCAP security is desired when using the Shared Lease Identifier feature, the confidentiality of the Lease Identifier MUST be maintained by encrypting all messages that contain it. A Lease Identifier that includes a long cryptographically random number (at least eight octets in length) MUST be used in this circumstance so that it is not easy to guess the Lease Identifier.

2.4.1. Random Lease Identifier

The first octet of a Random Lease Identifier is the Lease Identifier type (0 to indicate Random Lease Identifier). After this come a sequence of octets, which SHOULD represent a long random number (at least 16 octets) from a decent random number generator.

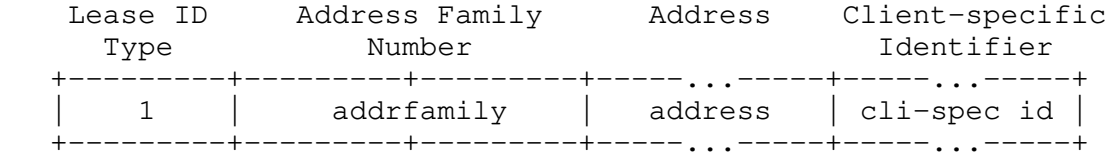
A Random Lease Identifier does not include any indication of its length. It is assumed that this may be determined by external means, such as a length field preceding the Lease Identifier.



2.4.2. Address-Specific Lease Identifier

The first octet of an Address-Specific Lease Identifier is the Lease Identifier type (1 to indicate Address-Specific Lease Identifier). After this comes a two octet IANA-defined address family number (including those defined in [10]), an address from the specified address family, and a client-specific identifier (such as a sequence number or the current time).

An Address-Specific Lease Identifier does not include any indication of its length. It is assumed that this may be determined by external means, such as a length field preceding the Lease Identifier.



2.5. Associating Client and Server Messages

Messages between clients and servers are associated with one another using the Lease Identifier and the xid field. As described in section 2.1.4, the client MUST choose an xid so that it is unlikely that the same combination of xid, msgtype, and Lease Identifier will be used for two different messages within [XID-REUSE-INTERVAL] (even across multiple clients which do not communicate among themselves). The Lease Identifier option, msgtype, and xid field MUST be included in each message sent by the client or the server.

The client MUST check the Lease Identifier option and xid field in each incoming message to ensure that they match the Lease Identifier and xid for an outstanding transaction. If not, the message MUST be ignored. The server MUST check the Lease Identifier option and xid field in each incoming message to establish the proper context for the message. If a server cannot process a message because it is invalid for its context, the server MUST generate and process an Invalid Request error, as described in section 2.6. A transaction can be an attempt to allocate a multicast address (consisting of DISCOVER, OFFER, REQUEST, ACK, and NAK messages), an attempt to renew a lease (consisting of RENEW, ACK, and NAK messages), an attempt to release a previously allocated multicast address (consisting of RELEASE, ACK, and NAK messages), or an attempt to acquire configuration parameters (consisting of GETINFO, ACK, and NAK messages).

2.6. Processing Errors

If a MADCAP server encounters an error while processing a message, there are two different ways to process this error. If it is clear that the message is not a NAK, the server SHOULD respond with a NAK containing the appropriate Error option. However, the server MAY decide to completely ignore chronic offenders. If the message is a NAK or it is not clear whether the message is a NAK (for instance, the message is garbled or has an incorrect version number), the server SHOULD ignore the message. This avoids NAK loops.

If a MADCAP client encounters an error while processing a message, it MUST ignore the message.

2.7. Multicast Scopes

RFC 2365 [3] provides for dividing the multicast address space into a number of administrative scopes. Routers should be configured so that each scope corresponds to a particular partition of the network into disjoint regions. Messages sent to a multicast address that falls within a certain administrative scope should only be delivered to hosts that have joined that multicast group *and* fall within the same region as the sender. For instance, packets sent to an address in the organization-local scope should only be delivered to hosts that have joined that group and fall within the same organization as the sender.

Different sets of scopes may be in effect at different places in the network and at different times. Before attempting to allocate an address from an administrative scope (other than global or link-level scope, which are always in effect), a MADCAP client SHOULD determine that the scope is in effect at its location at this time. Several techniques that a MADCAP client may use to determine the set of administrative scopes in effect (the scope list) are: manual configuration or configuration via MADCAP (using the Multicast Scope List option).

If a MADCAP client is unable to determine its scope list using one of these techniques, it MAY temporarily assume a scope list consisting of a single scope. If it is using IPv4, it SHOULD use IPv4 Local Scope (239.255.0.0/16), with a maximum TTL of 16. If it is using IPv6, it SHOULD use SCOP 3, with a maximum hop count of 16. Using this temporary scope list, it MAY attempt to contact a MADCAP server that can provide a scope list for it.

When a MADCAP client requests an address with a DISCOVER or REQUEST message, it MUST specify the administrative scope from which the address should be allocated. This scope is indicated with the Multicast Scope option. Likewise, the server MUST include the Multicast Scope option in all OFFER messages and all ACK messages sent in response to REQUEST messages.

2.8. Multicast TTL

Another way to limit propagation of multicast messages is by using TTL scoping. This technique has several disadvantages in comparison to administratively scoped multicast addresses (as described in [3]), but it is currently in widespread usage.

With TTL scoping, areas of the network are designated as scopes. Routers on the edges of these areas are configured with TTL thresholds so that multicast packets are not forwarded unless their

remaining TTL exceeds this threshold. A packet which should be restricted to a given TTL scope should have an initial TTL less than that scope's TTL threshold. Similar techniques may be used with IPv6, using the Hop Count field instead of the TTL field.

MADCAP may be used in an environment where administrative scoping is not in use and TTL scoping is. Under these circumstances, a MADCAP server MAY return a scope list that includes scopes with TTLs less than 255. The MADCAP client MAY then allocate addresses from these scopes, but MUST NOT set the TTL field of any packet sent to such an address to a value greater than the maximum TTL indicated in the scope list. In such an environment, it is recommended that the MADCAP Server Multicast Addresses associated with the IPv4 Local Scope (or SCOP 3 for IPv6) be configured using TTL thresholds so that packets sent to these addresses with TTL of 16 are not sent outside an appropriate boundary. This will allow MADCAP clients to use their default behavior for finding MADCAP servers.

In an environment where administrative scoping is in use, the maximum TTLs in the scope list SHOULD be 255. The admin scope zone boundary routers will prevent leakage of MADCAP packets beyond appropriate limits.

2.9. Locating MADCAP Servers

There are several ways for a MADCAP client to locate a MADCAP server. For instance, the client may be configured with an IP address.

The RECOMMENDED technique is for the client to send an GETINFO message to a MADCAP Server Multicast Address and wait for ACK responses. This technique is described in more detail in the next section.

2.10. MADCAP Server Multicast Address

Each multicast scope has an associated MADCAP Server Multicast Address. This address has been reserved by the IANA as the address with a relative offset of -1 from the last address of a multicast scope.

A MADCAP client looking for servers that can provide multicast allocation services MAY send an GETINFO message to a MADCAP Server Multicast Address. Any MADCAP servers listening to this address SHOULD respond with a unicast ACK message to the client if they wish to offer a response.

The MADCAP Server Multicast Address used by a client MAY be established by configuration. If a client has no such configuration, it SHOULD use the MADCAP Server Multicast Address associated with IPv4 Local Scope (or SCOP 3 for IPv6) with maximum TTL of 16, unless otherwise configured.

2.11. Going Beyond the Local Scope

If a client receives no response to a message sent to a MADCAP Server Multicast Address (after retransmission), it MAY send the message to a larger scope and repeat this process as necessary. However, the client MUST NOT send a MADCAP message to the MADCAP Server Multicast Address associated with the global scope.

This technique allows MADCAP servers to provide services for scopes in which they do not reside. However, this is a dangerous and complicated technique and is NOT RECOMMENDED at this time. Therefore, MADCAP clients SHOULD only send multicast messages to the MADCAP Server Multicast Address corresponding to the IPv4 Local Scope (or SCOP 3, if using IPv6), unless configured otherwise.

MADCAP servers that wish to provide services for scopes in which they do not reside MUST make special efforts to ensure that their services meet clients' needs for largely conflict-free allocation and accurate scope list information. In particular, coordinating with other servers that provide services for this scope may be difficult. Also, establishing which scope the client is in may be difficult. If a MADCAP server is not prepared to provide services for scopes in which it does not reside, it SHOULD ignore DISCOVER and REQUEST messages whose scope does not match or enclose the scope of the MADCAP Server Multicast Address on which the request was received. It SHOULD also ignore GETINFO messages that are not received on the MADCAP Server Multicast Address for IPv4 Local Scope.

2.12. Clock Skew

The Current Time option is used to detect and handle clock skew between MADCAP clients and servers. This option MUST be included in any MADCAP message that includes an absolute time (such as the Start Time option). It MAY be included in any DISCOVER, OFFER, REQUEST, RENEW, or ACK message.

Clock skew is a situation where two systems have clocks that are not synchronized. Many protocols (such as DHCP) ignore clock skew by using relative times. MADCAP could use a similar technique, but this leads to nasty situations due to the way multicast addresses are used.

For example, assume that at 1 PM UTC a client whose clock is one hour fast requests a lease for one hour starting in one hour. If we were using relative times for MADCAP, the server, whose clock is set correctly, would reserve a multicast address for 2 to 3 PM UTC and grant the request. If the client was the only one using the lease, everything would be OK. The client would start using the lease in one hour and continue for one hour. This would coincide with the time the server had reserved (although the client would think it was 3 to 4 PM UTC).

However, multicast addresses are usually used by several parties at once. The client would probably use SAP (or some other mechanism for conveying SDP) to advertise a session using the multicast address just leased. SDP uses absolute times, since it may be sent via email, web, or other store-and-forward mechanisms. So the client would advertise the session as running from 3 to 4 PM UTC. Any clients whose clocks are set correctly would use the address during this interval. Since the server only reserved the address from 2 to 3 PM UTC, this might cause the address to be used for multiple sessions simultaneously.

MADCAP cannot solve all clock skew problems. That is the domain of NTP [4]. However, it does attempt to detect substantial clock skew between MADCAP clients and servers so that this clock skew does not cause massive collisions in multicast address usage later on.

The Current Time option contains the sender's opinion of the current time in UTC at or about the time the message was assembled. Because of delays in transmission and processing, this value will rarely match the receiver's opinion of the current time at the time the option is processed by the receiver. However, difference greater than a minute or two probably indicate clock skew between the sender and the receiver.

MADCAP servers SHOULD expect and tolerate a small amount of clock skew with their clients by ensuring that multicast addresses are allocated for an extra period of time [EXTRA-ALLOCATION-TIME] on either side of the lease given to the client. However, large amounts of clock skew require special handling. The value of [EXTRA-ALLOCATION-TIME] MUST be a configurable parameter, since local circumstances may vary. The RECOMMENDED default is one hour.

However, large amounts of clock skew will cause problems later when sessions are advertised. If a MADCAP server detects clock skew greater than [CLOCK-SKEW-ALLOWANCE], it MUST generate and process an Excessive Clock Skew error, as described in section 2.6. The server MAY also log a message. The value of [CLOCK-SKEW-ALLOWANCE] MUST be a configurable parameter, since local circumstances may vary. The

RECOMMENDED default is 30 minutes.

2.13. Optional Features

Each MADCAP client or server MAY implement one or more optional features. Optional features of MADCAP are identified with a two octet feature code.

A MADCAP client MAY request, require, or indicate support for an optional feature by including a Feature List option in a message. For more information about optional features, see the description of the Feature List option.

Table 4 lists the feature codes defined at this time and sections 2.13.1 and 2.13.2 describe how these features work.

New MADCAP feature codes may only be defined by IETF Consensus, as described in section 5.

Feature Code	Feature Name
0	Server Mobility
1	Retry After
2	Shared Lease Identifier

Table 4: MADCAP Feature Codes

2.13.1. Server Mobility

The Server Mobility feature allows an address allocated on one MADCAP server to be renewed or released on a different MADCAP server. This requires communication and coordination among MADCAP servers. The primary benefits are immunity to the failure of a single MADCAP server and perhaps greater performance through load balancing.

In order to take advantage of the Server Mobility feature, a MADCAP client must ensure that the feature is implemented by both the server that is used for the original allocation and the server that is used for the renewal or release. The best way to ensure this is to include the Server Mobility feature in the required list of a Feature List option in the REQUEST message used to allocate the address (and the DISCOVER message, if one is used). When the time comes to renew or release the address, the client SHOULD send a unicast RENEW or RELEASE message to the server from which it allocated the address. However, if this server is unavailable, the client MAY send the RENEW or RELEASE message to any other server that includes the Server Mobility feature in its list of supported features. The client can find such a server by (for instance) sending an GETINFO message with

an Option Request List option that includes the Feature List option code.

If the MADCAP client does not want to require this feature when allocating addresses, it may include the feature in the requested list of a Feature List option and see if the server includes the feature in the required list of a Feature List option in the ACK message.

Even if the Server Mobility feature is used, there is no guarantee that a server will be available to perform the renewal or release or that the renewal or release will succeed. Server connectivity may have failed, for instance.

2.13.2. Retry After

The Retry After feature allows a MADCAP server to ask the MADCAP client to retry its request later, as may be required when allocating large numbers of addresses or allocating addresses for a long period of time.

For instance, if a MADCAP client requests 1000 addresses, administrative approval may be required or allocation of more addresses from another MASC domain may be necessary. This may take several hours or several days. If the MADCAP client and server both support the Retry After feature, the MADCAP server can send back an ACK message with a Retry Time option indicating when the addresses may be ready. The client can retry its request after the Retry Time to get the addresses.

If a MADCAP client includes the Retry After feature in the supported list of a Feature List option in a REQUEST message, a MADCAP server that supports the Retry After feature MAY decide to begin a lengthy allocation process. In this case, the MADCAP server will include an empty List of Address Ranges option in its ACK message, a Feature List option that includes the Retry After feature in the required list, and a Retry Time option with a time after which the client should retry the REQUEST.

The client MUST NOT include the Retry After feature in the requested or required list of a Feature List option, since the decision about whether Retry After is desirable should be left to the MADCAP server.

At some later time (preferably after the time indicated in the Retry Time option), the client SHOULD send a REQUEST message with all the same options as the original REQUEST message (especially the Lease Identifier option), but with a new xid value. The server MAY return a normal ACK or NAK message at this point or it MAY continue the transaction to a later time by including an empty List of Address Ranges option in its ACK message, a Feature List option that includes the Retry After feature in the required list, and a Retry Time option with a later time after which the client should retry the REQUEST.

At any point after receiving the initial ACK message with the Retry Time option, the client MAY terminate the allocation process and any accompanying lease by sending to the server performing the allocation (or another server if the Server Mobility feature is also in effect) a RELEASE message with the Lease Identifier included in the original REQUEST message.

The Retry After feature may also be used when renewing a lease. In this case, the description above applies except that the client sends a RENEW message instead of a REQUEST message.

If a client sends a RENEW message with a Lease Identifier that matches a lease which is currently undergoing allocation with the Retry After feature in response to a REQUEST message, the server MUST generate and process an Invalid Request error in the manner described in section 2.6. Also, if a client sends a RENEW message with a Lease Identifier that matches a lease which is currently undergoing allocation with the Retry After feature in response to a RENEW message, but the options supplied with the two RENEW messages do not match, the server MUST generate and process an Invalid Request error in the manner described in section 2.6.

Note that the Retry After feature may complicate the application API. For this reason, a MADCAP client may request the Retry After feature for some messages and not for others. This should not cause problems for a robust MADCAP server. In general, servers should not expect consistent behavior from clients except as required by this specification. This also applies to clients' expectations.

2.13.3. Shared Lease Identifier

For conferencing applications, it may be desirable to allow conference participants to modify a lease used for the conference. The Shared Lease Identifier feature code is used to support this requirement.

If this feature code was requested by the client and implemented by the server when the lease was allocated, the server SHOULD disable any authentication requirements pertaining to this lease, allowing any client that knows the Lease Identifier to modify the lease.

A MADCAP client wishing to use the Shared Lease Identifier feature should include this feature in the requested or required lists of the Feature List option of a REQUEST message when first allocating the lease. If the feature was required, the server SHOULD try to implement it for this request and include the feature in the required list of the response. If the server can not implement the feature for this request, it MUST generate and process a Required Feature Not Supported error in the manner described in section 2.6. If the feature was requested, the server SHOULD try to implement the feature and include the feature in the required list of the response. However, if the server cannot implement the feature, it may simply skip it.

Subsequent requests pertaining to a lease for which the Shared Lease Identifier feature was implemented at allocation time MAY include the Shared Lease Identifier feature in the requested or required lists of the Feature List option. In this case, the server SHOULD try to implement the feature by disabling any authentication requirements pertaining to this lease, allowing any client that knows the Lease Identifier to modify the lease, and including the feature in the required list of the response. If the server cannot implement the feature, it SHOULD skip it if the feature was requested. But if the feature was required and the server cannot implement it, the server MUST generate and process a Required Feature Not Supported error in the manner described in section 2.6.

3. MADCAP Options

As described earlier, each MADCAP message includes an options field consisting of a list of tagged parameters that are called "options". All options consist of a two octet option code and a two octet option length, followed by the number of octets specified by the option length.

This section defines a set of option codes for use in MADCAP messages. New options may be defined using the process defined in section 5. The options are listed in numerical order.

Table 5 summarizes which options are allowed with each message type.

Option	GETINFO	ACK (in response to GETINFO)
Lease Time	MUST NOT	MUST NOT
Server Identifier	MUST NOT	MUST
Lease Identifier	MUST	MUST
Multicast Scope	MUST NOT	MUST NOT
Option Request List	MUST	MUST NOT
Start Time	MUST NOT	MUST NOT
Number of Addresses Requested	MUST NOT	MUST NOT
Requested Language	MAY	MUST NOT
Multicast Scope List	MUST NOT	MAY
List of Address Ranges	MUST NOT	MUST NOT
Current Time	MUST NOT	MAY
Feature List	MAY	MAY
Retry Time	MUST NOT	MUST NOT
Minimum Lease Time	MUST NOT	MUST NOT
Maximum Start Time	MUST NOT	MUST NOT
Error	MUST NOT	MUST NOT

Option	DISCOVER	OFFER
Lease Time	MAY	MUST
Server Identifier	MUST NOT	MUST
Lease Identifier	MUST	MUST
Multicast Scope	MUST	MUST
Option Request List	MUST NOT	MUST NOT
Start Time	MAY	MAY
Number of Addresses Requested	MAY	MUST NOT
Requested Language	MUST NOT	MUST NOT
Multicast Scope List	MUST NOT	MUST NOT
List of Address Ranges	MAY	MAY
Current Time	MAY	MAY
Feature List	MAY	MAY
Retry Time	MUST NOT	MUST NOT
Minimum Lease Time	MAY	MUST NOT
Maximum Start Time	MAY	MUST NOT
Error	MUST NOT	MUST NOT

Option	REQUEST	ACK (in response to REQUEST)
Lease Time	MAY	MUST
Server Identifier	MUST (if multicast)	MUST
Lease Identifier	MUST	MUST
Multicast Scope	MUST	MUST
Option Request List	MUST NOT	MUST NOT
Start Time	MAY	MAY
Number of Addresses Requested	MAY	MUST NOT
Requested Language	MUST NOT	MUST NOT
Multicast Scope List	MUST NOT	MUST NOT
List of Address Ranges	MAY	MUST
Current Time	MAY	MAY
Feature List	MAY	MAY
Retry Time	MUST NOT	MAY
Minimum Lease Time	MAY	MUST NOT
Maximum Start Time	MAY	MUST NOT
Error	MUST NOT	MUST NOT

Option	RENEW	ACK (in response to RENEW)
Lease Time	MAY	MUST
Server Identifier	MUST NOT	MUST
Lease Identifier	MUST	MUST
Multicast Scope	MUST NOT	MUST
Option Request List	MUST NOT	MUST NOT
Start Time	MAY	MAY
Number of Addresses Requested	MUST NOT	MUST NOT
Requested Language	MUST NOT	MUST NOT
Multicast Scope List	MUST NOT	MUST NOT
List of Address Ranges	MUST NOT	MUST
Current Time	MAY	MAY
Feature List	MAY	MAY
Retry Time	MUST NOT	MAY
Minimum Lease Time	MAY	MUST NOT
Maximum Start Time	MAY	MUST NOT
Error	MUST NOT	MUST NOT

Option	RELEASE	ACK (in response to RELEASE)
Lease Time	MUST NOT	MUST NOT
Server Identifier	MUST NOT	MUST
Lease Identifier	MUST	MUST
Multicast Scope	MUST NOT	MUST NOT
Option Request List	MUST NOT	MUST NOT
Start Time	MUST NOT	MUST NOT
Number of Addresses Requested	MUST NOT	MUST NOT
Requested Language	MUST NOT	MUST NOT
Multicast Scope List	MUST NOT	MUST NOT
List of Address Ranges	MUST NOT	MUST NOT
Current Time	MUST NOT	MUST NOT
Feature List	MAY	MAY
Retry Time	MUST NOT	MUST NOT
Minimum Lease Time	MUST NOT	MUST NOT
Maximum Start Time	MUST NOT	MUST NOT
Error	MUST NOT	MUST NOT

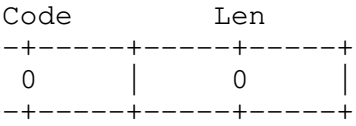
Option	NAK
Lease Time	MUST NOT
Server Identifier	MUST
Lease Identifier	MUST
Multicast Scope	MUST NOT
Option Request List	MUST NOT
Start Time	MUST NOT
Number of Addresses Requested	MUST NOT
Requested Language	MUST NOT
Multicast Scope List	MUST NOT
List of Address Ranges	MUST NOT
Current Time	MUST NOT
Feature List	MAY
Retry Time	MUST NOT
Minimum Lease Time	MUST NOT
Maximum Start Time	MUST NOT
Error	MUST

Table 5: Options allowed in MADCAP messages

3.1. End

The End option marks the end of valid information in the options field. This option MUST be included at the end of the options field in each MADCAP message.

The code for this option is 0, and its length is 0.

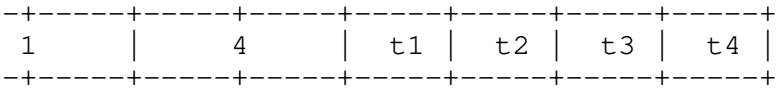


3.2. Lease Time

This option is used in a client request (DISCOVER, REQUEST, or RENEW) to allow the client to request a lease time for the multicast address. In a server reply (OFFER or ACK), a MADCAP server uses this option to specify the lease time it is willing to offer.

The time is in units of seconds, and is specified as a 32-bit unsigned integer.

The code for this option is 1, and its length is 4.



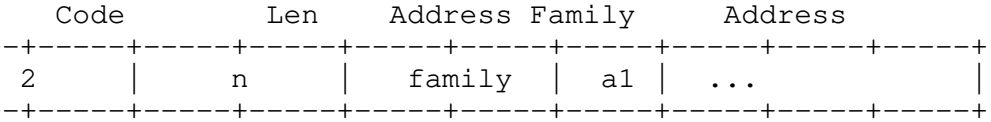
3.3. Server Identifier

This option contains the IP address of a MADCAP server. A two octet address family number (as defined by IANA, including those defined in [10]) is stored first, followed by the address. The address family for this address is not determined by the addrfamily field in the fixed header so that addresses from one family may be allocated while communicating with a server via addresses of another family.

All messages sent by a MADCAP server MUST include a Server Identifier option with the IP address of the server sending the message.

MADCAP clients MUST include a Server Identifier option in multicast REQUEST messages in order to indicate which OFFER message has been accepted.

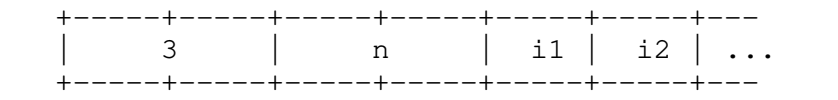
The code for this option is 2, and its minimum length is 3.



3.4. Lease Identifier

This option is used by MADCAP clients to specify a unique lease identifier. For more information about this option and how it is used, see section 2.4.

The code for this option is 3, and its minimum length is 1.

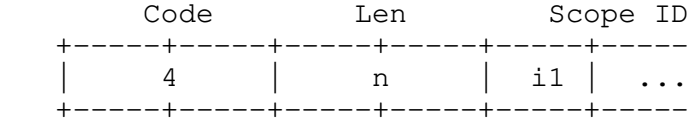


3.5. Multicast Scope

The multicast scope option is used by the client to indicate the requested multicast scope in a DISCOVER or REQUEST message. It is also used by the MADCAP server to indicate the scope of an assigned address.

The client may obtain the scope list through the Multicast Scope List option or using some other means. The Scope ID is the first multicast address in the scope. The address family of the Scope ID is determined by the addrfamily field in the fixed header.

The code for this option is 4, and its minimum length is 1.

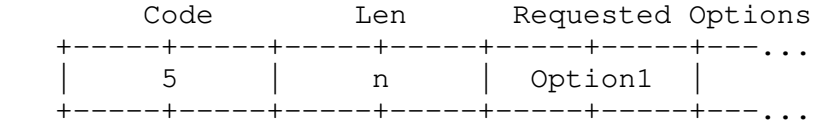


3.6. Option Request List

This option is used by a MADCAP client in an GETINFO message to request that certain options be included in the server's ACK response. The server SHOULD try to include the specified options in its response, but is not required to do so.

The format of this option is a list of option codes.

The code for this option is 5 and the minimum length is 2.



3.7. Start Time

The Start Time option specifies the starting time for a multicast address lease.

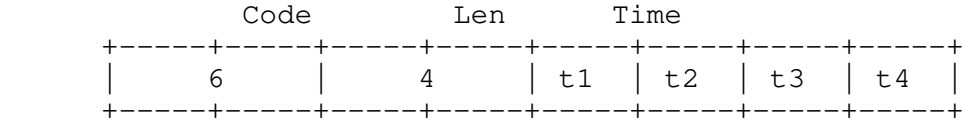
A client may include this option in a DISCOVER, RENEW, or REQUEST message to request a multicast address for use at a future time. A server may include this option in an OFFER message or in an ACK in response to REQUEST or RENEW message to indicate that a lease has been granted which starts at a future time.

If the Start Time option is present, the IP Address Lease Time option specifies the duration of the lease beginning at the Start Time option value.

If the Start Time option is present, the Current Time option MUST also be present, as described in section 2.12.

The time value is an unsigned 32 bit integer in network byte order giving the number of seconds since 00:00 UTC, 1st January 1970. This can be converted to an NTP timestamp by adding decimal 2208988800. This time format will not wrap until the year 2106.

The code for this option is 6 and the length is 4.



3.8. Number of Addresses Requested

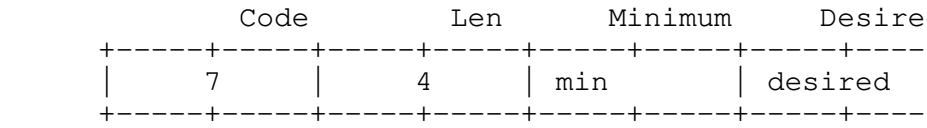
This option specifies the minimum and desired number of addresses requested by the client. It is only used in DISCOVER and REQUEST messages and is only sent by the client.

The minimum and desired number of addresses requested are unsigned 16 bit integers in network byte order. The minimum MUST be less than or equal to the desired number. If a message is received where this is not the case, the MADCAP server MUST generate and process an Invalid Request error in the manner described in section 2.6.

The client MAY obtain more than one address either by repeating the protocol for every address or by requesting several addresses at the same time via this option. When the client is requesting only one address, this option SHOULD NOT be included. A MADCAP server

receiving a DISCOVER or REQUEST packet including this option MUST include between the minimum and desired number of addresses in any OFFER or ACK response.

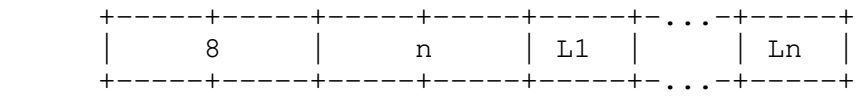
The code for this option is 7 and the length is 4.



3.9. Requested Language

This option specifies the language in which the MADCAP client would like strings such as zone names to be returned. It is only included in an GETINFO message sent by the client. It is an RFC 1766 [6] language tag. The proper way to handle this tag with respect to zone names is discussed further in the definition of the Multicast Scope List option.

The code for this option is 8 and the minimum length is 0.



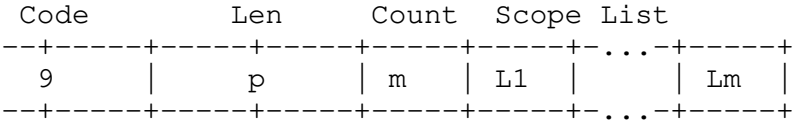
3.10. Multicast Scope List

This option is sent by the server in an ACK message in response to an GETINFO message sent by the client.

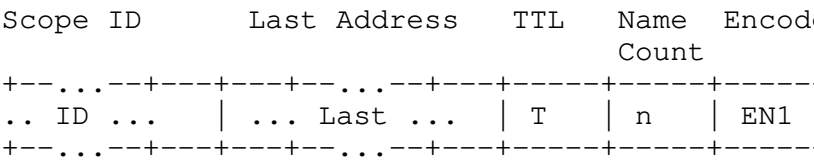
If the client did not include a Requested Language option in its GETINFO message, the MADCAP server SHOULD return all zone names for each zone. If the client included a Requested Language option in its GETINFO message, the MADCAP server MUST return no more than one zone name for each zone. For each zone, the MADCAP server SHOULD first look for a zone name that matches the requested language tag (using a case-insensitive ASCII comparison). If any names match, one of them should be returned. Otherwise, the MADCAP server SHOULD choose another zone name to return (if any are defined). It SHOULD give preference to zone names that are marked to be used if no name is available in a desired language.

The code for this option is 9 and the minimum length is 0.

The format of the multicast scope list option is:

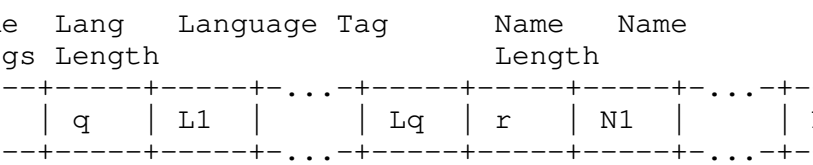


The scope list is a list of m tuples, where each tuple is of the form:



where Scope ID is the first multicast address in the scope, Last Address is the last multicast address in the scope, TTL is the multicast TTL value for the multicast addresses of the scope, and Name Count is the number of encoded names for this zone (which may be zero). The address family of the Scope ID and Last Address is determined by the addrfamily field in the fixed header. Note that a particular MADCAP server may be allocating addresses out of some subset of the scope. For instance, the addresses in the scope may be divided among several servers in some way.

Each encoded name is of the form



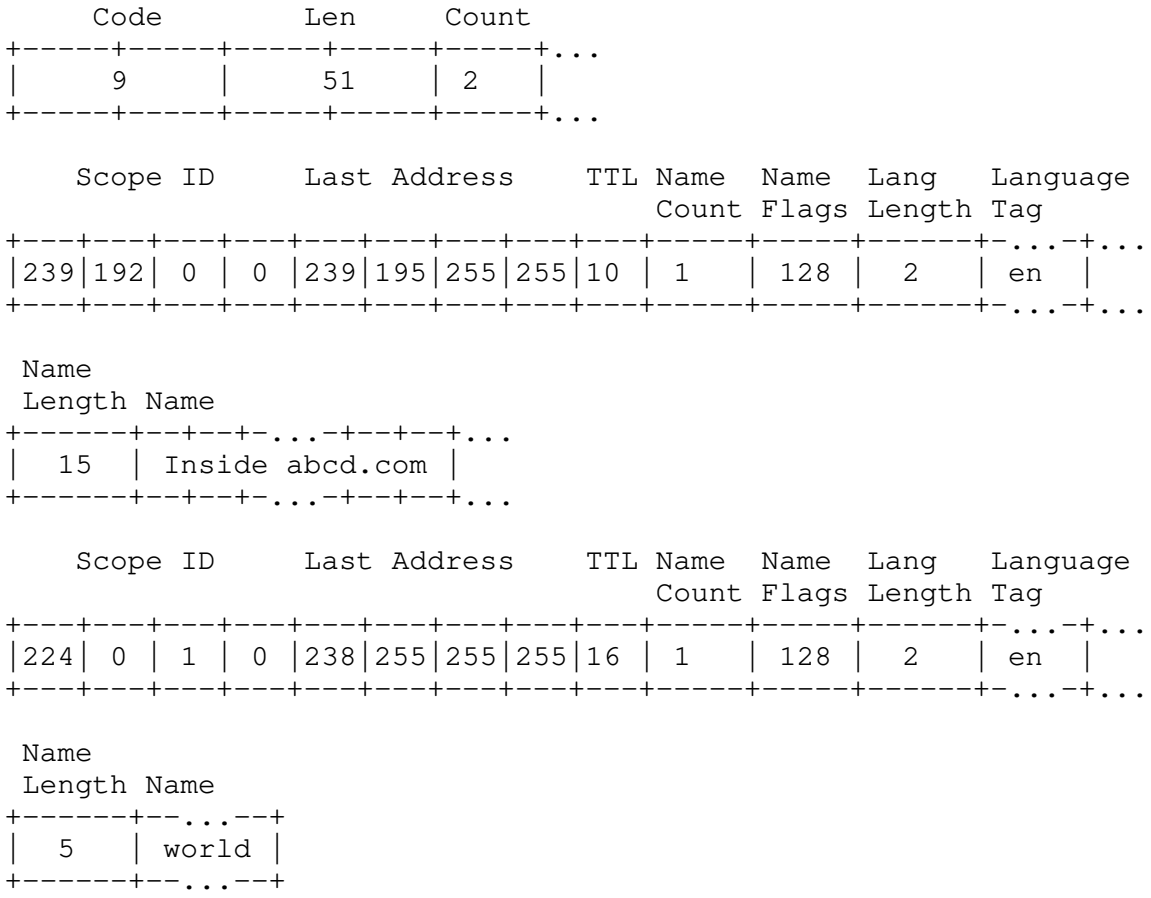
where Name Flags is a flags field with flags defined below, Lang Length is the length of the Language Tag in octets (which MUST NOT be zero), Language Tag is a language tag indicating the language of the zone name (as described in [6]), Name Length is the length of the Name in octets (which MUST NOT be zero), and Name is a UTF-8 [5] string indicating the name given to the scope zone.

The high bit of the Name Flags field is set if the following name should be used if no name is available in a desired language. Otherwise, this bit is cleared. All remaining bits in the octet SHOULD be set to zero and MUST be ignored.

The Scope IDs of entries in the list MUST be unique and the scopes SHOULD be listed from smallest (topologically speaking) to largest. This makes it easier to display a consistent user interface, with scopes usually keeping the same order. However, scopes may not be strictly nested. In this circumstance, there is no strict ordering from smallest to largest and the server must use another technique for ordering the scope list.

Example:

There are two scopes supported by the multicast address allocation server: Inside abcd.com with addresses 239.192.0.0-239.195.255.255, and world with addresses 224.0.1.0-238.255.255.255. Then this option will be given as:



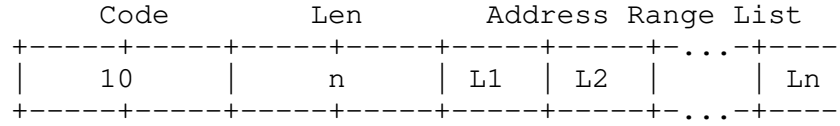
3.11. List of Address Ranges

This option is used by the server to provide the list of all the address ranges allocated to the client.

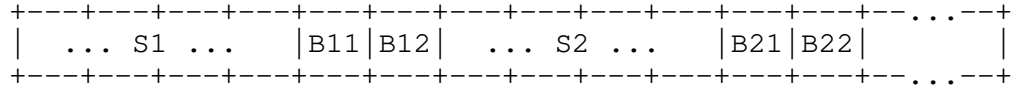
This option is also used by the client when requesting a lease for a specific set of addresses. This feature should be needed only rarely, such as when a lease is accidentally allowed to expire and it needs to be reallocated.

The address family of the addresses is determined by the addrfamily field.

The code for this option is 10 and the minimum length is 0.



where the Address Range List is of the following format.

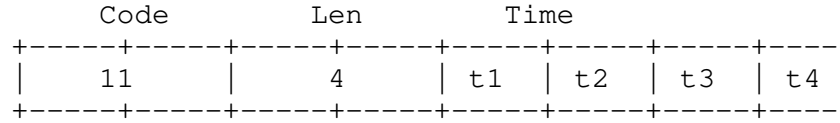


3.12. Current Time

This option is used to express what the sender thinks the current time is. This is useful for detecting clock skew. This option MUST be included if the Start Time or Maximum Start Time options are used, as described in section 2.12.

The time value is an unsigned 32 bit integer in network byte order giving the number of seconds since 00:00 UTC, 1st January 1970. This can be converted to an NTP [4] timestamp by adding decimal 2208988800. This time format will not wrap until the year 2106.

The code for this option is 11 and the length is 4.



3.13. Feature List

This option lists optional MADCAP features supported, requested, or required, by the sender. This option MAY be included in any message sent by a MADCAP server or client.

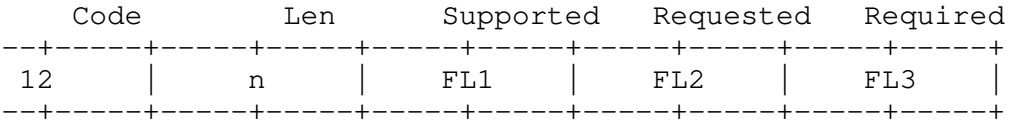
Optional features of MADCAP are identified with a two octet feature code. New MADCAP feature codes may only be defined by IETF Consensus, as described in section 5.

The Feature List option consists of three separate lists: supported features, requested features, and required features. Each list consists of an unordered list of feature codes. The supported list is used by MADCAP clients or servers to indicate the features that the sender supports. The requested and required lists are used by MADCAP clients to indicate which features are requested of or required from a MADCAP server. The required list is used by MADCAP servers to indicate which features were implemented by the MADCAP server in processing this message. Messages sent by MADCAP servers MUST NOT include any feature codes in the requested list.

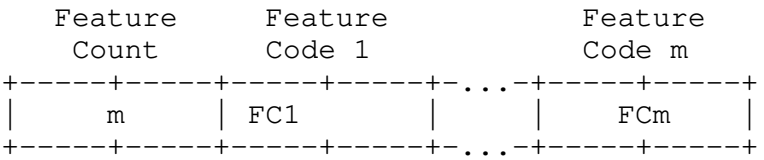
If a MADCAP client includes the Feature List option in a message, it MAY include features in any of the lists: supported, requested, and required. If a MADCAP server receives a message containing the Feature List option and it does not support all of the features in the required list, it MUST generate and process a Required Feature Not Supported error in the manner described in section 2.6. If the server supports all of the features in the required list, it MUST implement them as appropriate for this message. It SHOULD try to implement the features in the requested list and it MAY implement any of the features in the supported list. If an optional feature (such as Retry After) is not included in any part of the Feature List option included in the client's message (or if the client does not include a Feature List option in its message), the server MUST NOT use that feature in its response.

If a MADCAP server does respond to a client's message that includes a Feature List option, the server MUST include a Feature List option with a supported features list that lists the features that it supports, a required features list that lists the features that it implemented in responding to this message (which must be included in the supported features list of the client's Feature List option), and an empty requested features list.

The code for this option is 12 and the minimum length is 6.



where each of the Feature Lists is of the following format:



3.14. Retry Time

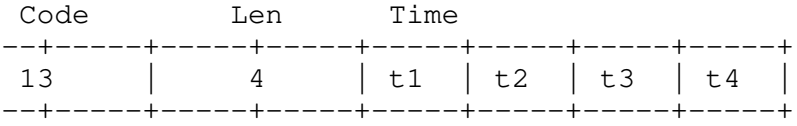
The Retry Time option specifies the time at which a client should retry a REQUEST or RENEW message when using the Retry After feature.

This option should only be sent by a MADCAP server in an ACK when responding to a REQUEST or RENEW message that includes the Retry After feature in the supported, requested, or required list. For more discussion of Retry After, see section 2.13.2.

If the Retry Time option is present, the Current Time option MUST also be present.

The time value is an unsigned 32 bit integer in network byte order giving the number of seconds since 00:00 UTC, 1st January 1970. This can be converted to an NTP timestamp by adding decimal 2208988800. This time format will not wrap until the year 2106.

The code for this option is 13 and the length is 4.

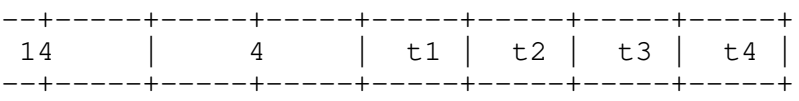


3.15. Minimum Lease Time

This option is used in a client request (DISCOVER, REQUEST, or RENEW) to allow the client to specify a minimum lease time for the multicast address. If a server cannot meet this minimum lease time, it MUST generate and process a Valid Request Could Not Be Completed error in the manner described in section 2.6.

The time is in units of seconds, and is specified as a 32-bit unsigned integer.

The code for this option is 14, and its length is 4.



3.16. Maximum Start Time

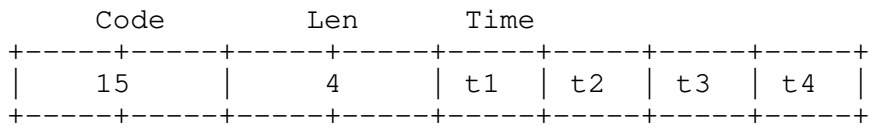
The Maximum Start Time option specifies the latest starting time that the client is willing to accept for a multicast address lease.

A client may include this option in a DISCOVER, RENEW, or REQUEST message to specify that it does not want to receive a lease with a starting time later than the specified value. If a server cannot meet this maximum start time, it MUST generate and process a Valid Request Could Not Be Completed error in the manner described in section 2.6.

If the Maximum Start Time option is present, the Current Time option MUST also be present, as described in section 2.12.

The time value is an unsigned 32 bit integer in network byte order giving the number of seconds since 00:00 UTC, 1st January 1970. This can be converted to an NTP timestamp by adding decimal 2208988800. This time format will not wrap until the year 2106.

The code for this option is 15 and the length is 4.



3.17. Error

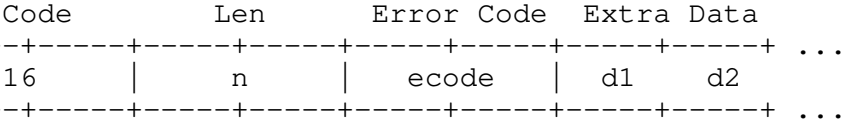
A MADCAP server includes this option in a NAK message to indicate why the request failed. A MADCAP server MUST include an Error option in each NAK message.

The first two octets of an Error option contain a MADCAP error code. Several MADCAP error codes are defined later in this section. New MADCAP error codes may only be defined by IETF Consensus, as described in section 5.

Any remaining octets in the Error option contain extra data about the error. The format of this data depends on the error code. The definition of a MADCAP error code must include a definition of the extra data to be included with that error code.

A client that receives a NAK message containing an Error option MAY log or display a message indicating the error code and extra data received. The client MUST NOT retransmit a message once a NAK response to that message has been received. The client MAY adjust the message to correct the error and send the corrected message or send a message to a different server.

The code for this option is 16, and the minimum length is 2.



3.17.1. Valid Request Could Not Be Completed

MADCAP error code 0 indicates that the request was valid, but could not be completed with the available addresses and the current configuration. The extra data is a two octet option code indicating which option caused the problem. A value of 0xFFFF indicates that the problem was not with a specific option.

3.17.2. Invalid Request

MADCAP error code 1 indicates that the request was malformed or invalid in some other manner. The extra data is a two octet option code indicating which option caused the problem. A value of 0xFFFF indicates that the problem was not with a specific option.

3.17.3. Excessive Clock Skew

MADCAP error code 2 indicates excessive clock skew (see section 2.12). The extra data consists of a four octet time value representing the server's idea of the current time, an unsigned 32 bit integer in network byte order giving the number of seconds since 00:00 UTC, 1st January 1970. This can be converted to an NTP timestamp by adding decimal 2208988800. This time format will not wrap until the year 2106.

3.17.4. Lease Identifier Not Recognized

MADCAP error code 3 indicates that the Lease Identifier was not recognized (usually in response to a RENEW or RELEASE message). There is no extra data.

3.17.5. Required Feature Not Supported

MADCAP error code 4 indicates that at least one feature included in the required list of the Feature List option is not supported. The extra data contains a list of the feature codes in the required list that are not supported.

3.17.6. Experimental Use

MADCAP error codes 1024-2047 are reserved for experimental use. The format of the extra data included with these error codes is not defined.

4. Security Considerations

MADCAP has relatively basic security requirements. At present there is no way of enforcing authorized use of multicast addresses in the multicast routing/management protocols. Therefore, it is not possible to identify unauthorized use of multicast address by an adversary. Moreover, a multicast address allocated to a user/system can be used by other systems without violating terms of the multicast address allocation. For example, a system may reserve an address to be used for a work group session where each and every member of the work group is allowed to transmit packets using the allocated group address. In other words, the multicast address allocation protocol does not dictate how the address should be used, it only dictates the time period for which it can be used and who gets to release it or renew it. When an address is allocated to a system/user, it basically means that no other user/system (most likely) will be allocated that address for the time period, without any restrictions on its use.

To protect against rogue MADCAP servers (mis-configured servers and intentional), clients in certain situations would like to authenticate the server. Similarly, for auditing or book-keeping purposes, the server may want to authenticate clients. Moreover, in some cases, the server may have certain policies in place to restrict the number of addresses that are allocated to a system or a user. This feature is of much value when a well behaved but naive user or client requests a large number of addresses, and therefore, inadvertently impacts other users or systems. Therefore, an administrator may want to exert a limited amount of control based on the client identification. The client identification could be based

on the system or user identity. In most practical situations, system identification will suffice, however, particularly in case of multi-user systems, at times, user identification will play an important role. Therefore, authentication capabilities based on user identification may be desirable. As usual, data integrity is a strong requirement and if not protected, can lead to many problems including denial of service attacks.

In the case of MADCAP, confidentiality is not a strong requirement. In most of the cases, at least when a multicast address is in use, an adversary will be able to determine information that was contained in the MADCAP messages. In some cases, the users/systems may want to protect information in the MADCAP messages so that an adversary is not able to determine relevant information in advance and thus, plan an attack in advance. For example, if an adversary knows in advance (based on MADCAP messages) that a particular user has requested a large number of address for certain time period and scope, he may be able to guess the purpose behind such request and target an attack. When the Shared Lease Identifier feature is used, preserving the confidentiality of MADCAP messages becomes more important. Also, there may be features added to the protocol in the future that may have stronger confidentiality requirements.

The IPSEC protocol [8] meets client/server identification and integrity protection requirements stated above, requires no modification to the MADCAP protocol, and leverages extensive work in IETF and industry. Therefore, when security is a strong requirement, IPSEC SHOULD be used for protecting all the unicast messages of MADCAP protocol. When IPSEC based security is in use, all the multicast packets except GETINFO MUST be dropped by the MADCAP server. The prevalent implementations of IPSEC support client identification in form of system identification and do not support user identification. However, when desired, IPSEC with appropriate API's may be required to support user identification.

5. IANA Considerations

This document defines several number spaces (MADCAP options, MADCAP message types, MADCAP Lease Identifier types, MADCAP features, and MADCAP error codes). For all of these number spaces, certain values are defined in this specification. New values may only be defined by IETF Consensus, as described in [7]. Basically, this means that they are defined by RFCs approved by the IESG.

6. Acknowledgments

The authors would like to thank Rajeev Byrisetty, Steve Deering, Peter Ford, Mark Handley, Van Jacobson, David Oran, Thomas Pfenning, Dave Thaler, Ramesh Vyaghrapuri and the participants of the IETF for their assistance with this protocol.

Much of this document is based on [1] and [2]. The authors of this document would like to express their gratitude to the authors of these previous works. Any errors in this document are solely the fault of the authors of this document.

7. References

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

[2] Alexander, S. and R. Droms, "DHCP Options and BOOTP Vendor Extensions", RFC 2132, March 1997.

[3] Meyer, D., "Administratively Scoped IP Multicast", BCP 23, RFC 2365, July 1998.

[4] Mills, D., "Network Time Protocol (Version 3) Specification, Implementation and Analysis", RFC 1305, March 1992.

[5] Yergeau, F., "UTF-8, a transformation format of ISO 10646", RFC 2279, January 1998.

[6] Alvestrand, H., "Tags for the Identification of Languages", RFC 1766, March 1995.

[7] Alvestrand, H. and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 2434, October 1998.

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

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

[10] Reynolds, J. and J. Postel, "Assigned Numbers", STD 2, RFC 1700, October 1994.

[11] Deering, S., "Host Extensions for IP Multicasting", STD 5, RFC 1112, August 1989.

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

[13] Hinden, R. and S. Deering, "IP Version 6 Addressing Architecture", RFC 2373, July 1998.

[14] Schulzrinne, H., Casner, S., Frederick, R. and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", RFC 1889, January 1996.

8. Authors' Addresses

Stephen R. Hanna
Sun Microsystems, Inc.
One Network Drive
Burlington, MA 01803

Phone: +1.781.442.0166
EMail: steve.hanna@sun.com

Baiju V. Patel
Intel Corp.
Mail Stop: AG2-201
5200 NE Elam Young Parkway
Hillsboro, OR 97124

Phone: 503 696 8192
EMail: baiju.v.patel@intel.com

Munil Shah
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

Phone: 425 703 3924
EMail: munils@microsoft.com

APPENDIX A: Examples

This appendix includes several examples of typical MADCAP protocol exchanges.

1. Multicast Scope List Discovery

In this example, a MADCAP client wants to determine the scope list in effect. The client is using IPv4, so it starts by multicasting an GETINFO packet to the MADCAP Server Multicast Address corresponding to IPv4 Local Scope. This packet includes the Lease Identifier option, an Option Request List including the Multicast Scope List option code, and a Requested Language option containing the string "en", since the client is configured to prefer the English language.

Two MADCAP servers respond by sending ACK messages. These ACK messages include the Lease Identifier option and xid supplied by the client, the server's Server Identifier, and the Multicast Scope List with one name per scope (the one that most closely matches the language tag "en").

The following figure illustrates this exchange.

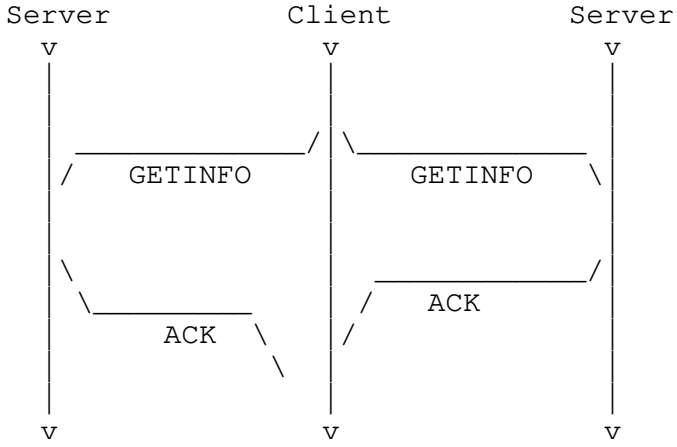


Figure 2: Timeline diagram of messages exchanged in Multicast Scope List Discovery example

2. Multicast Discovery and Address Allocation

In this example, the MADCAP client wants to allocate a multicast address from the global scope for use during the next two hours.

The client begins by multicasting a DISCOVER packet to the MADCAP Server Multicast Address associated with IPv4 Local Scope. This packet includes the Lease Time, Lease Identifier, and Multicast Scope options.

Any servers that receive the DISCOVER packet and can satisfy this request temporarily reserve an address for the client and unicast an OFFER packet to the client. These packets contain the Lease Time, Server Identifier, Lease Identifier, and Multicast Scope options.

After an appropriate delay, the client multicasts a REQUEST packet to the MADCAP Server Multicast Address. This packet contains all of the options included in the DISCOVER packet, but also includes the Server Identifier option, indicating which server it has selected for the request.

The server whose Server Identifier matches the one specified by the client responds with an ACK packet containing the options included in the OFFER packet, as well as a List of Address Ranges option listing the address allocated. All the other servers that had sent OFFER packets stop reserving an address for the client and forget about the whole exchange.

The client now has a two hour "lease" on the multicast address.

If the client had not received an ACK from the server, it would have retransmitted its REQUEST packet for a while. If it still received no response, it would start over with a new DISCOVER message.

The following figure illustrates this exchange.

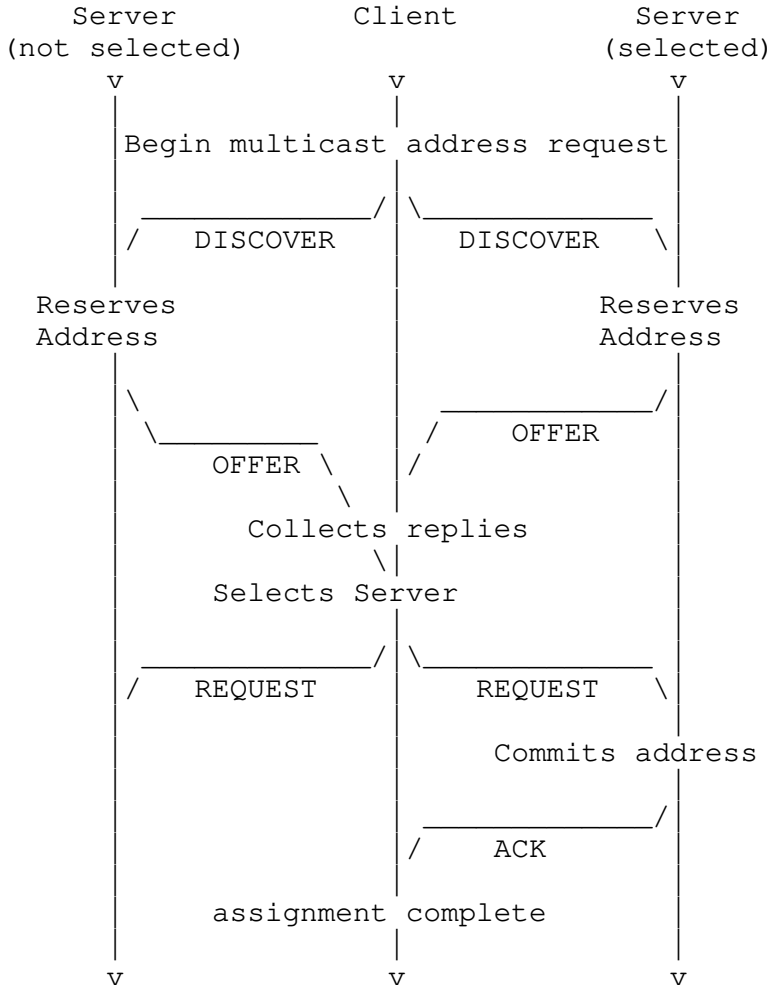


Figure 3: Timeline diagram of messages exchanged in Multicast Address Allocation example

3. Lease Extension

This is a continuation of the previous example. The client has already allocated a multicast address from the global scope for use during the next two hours. Half way through this two hour period, it decides that it wants to extend its lease for another hour.

The client unicasts a RENEW packet to the server from which it allocated the address. This packet includes the Lease Time and Lease Identifier options. The Lease Identifier matches the one used for the original allocation. The time included in the Lease Time is two

hours, since the client wants the lease to expire two hours from the current time.

The server responds with an ACK packet indicating that the lease extension has been granted. This packet includes the Lease Time, Server Identifier, Lease Identifier, Multicast Scope, and List of Address Ranges options.

If the server did not want to grant the requested lease extension, it would have responded with a NAK packet with the Lease Identifier option.

The following figure illustrates this exchange.

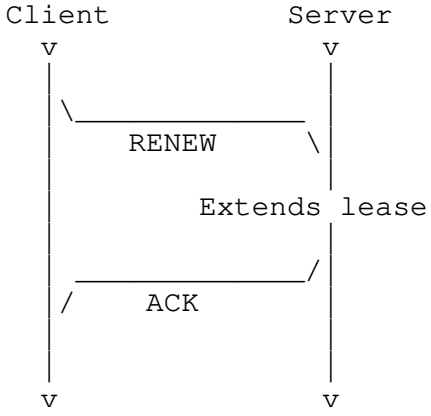


Figure 4: Timeline diagram of messages exchanged in Lease Extension example

4. Address Release

This is a continuation of the previous example. The client has already allocated a multicast address and extended its lease for another two hours. Half an hour later, the client finishes its use of the multicast address and wants to release it so it can be reused.

The client unicasts a RELEASE packet to the server from which it allocated the address. This packet includes the Lease Identifier option. The Lease Identifier matches the one used for the original allocation. When the server receives this packet, it cancels the client's lease on the address and sends an ACK packet to the client indicating that the lease has been released. This packet includes the Server Identifier and Lease Identifier options.

The following figure illustrates this exchange.

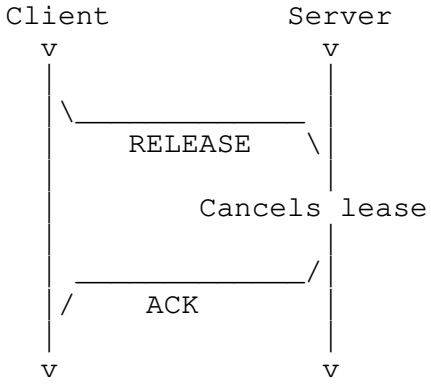


Figure 5: Timeline diagram of messages exchanged in Address Release example

5. Unicast Address Allocation

This is a continuation of the previous example. At some later time, the client decides to allocate another multicast address. Since it has recently worked with a server, it decides to try sending a unicast REQUEST to that server. If this doesn't work, it can always try a multicast DISCOVER, as illustrated in example 2.

The client unicasts a REQUEST packet to the server from which it wants to allocate the address. This packet includes the Lease Time, Lease Identifier, and Multicast Scope options.

The server responds with an ACK packet containing the Lease Time, Lease Identifier, and Multicast Scope options from the REQUEST packet, as well as the Server Identifier option and a List of Address Ranges option listing the address allocated.

The client now has a lease on the multicast address.

If the client had not received an ACK from the server, it would have retransmitted its REQUEST packet for a while. If it still received no response, it would start over with a multicast DISCOVER message.

The following figure illustrates this exchange.

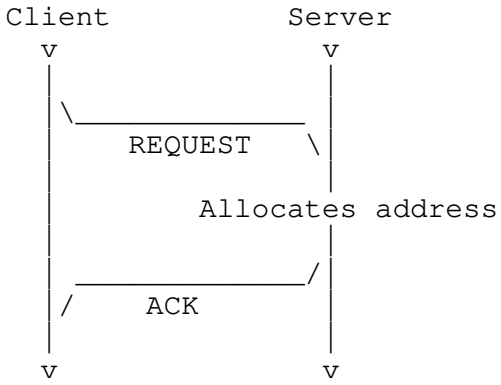


Figure 6: Timeline diagram of messages exchanged in Unicast Address Allocation example

APPENDIX B: Recommended Constant Values

Table 6 lists recommended values for constants defined in this specification.

Constant Name	Recommended Value
[CLOCK-SKEW-ALLOWANCE]	30 minutes
[DISCOVER-DELAY]	current retransmit delay
[EXTRA-ALLOCATION-TIME]	1 hour
[NO-RESPONSE-DELAY]	60 seconds
[OFFER-HOLD]	at least 60 seconds
[RESPONSE-CACHE-INTERVAL]	at least 60 seconds (5 minutes maximum)
[XID-REUSE-INTERVAL]	10 minutes (required)

Table 6: Recommended Constant Values

Full Copyright Statement

Acknowledgement

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