IMAP ANNOTATEMORE Extension
draft-daboo-imap-annotatemore-05

Status of this Memo

Copyright Notice

Abstract

The ANNOTATEMORE extension to the Internet Message Access Protocol
permits clients and servers to maintain "metadata" on IMAP4 servers.
It is possible to store data on a per-mailbox basis or on the server
as a whole.

Change History (to be removed prior to publication as an RFC)

Changes from -04 to -05:
1. Fix for valid IMAP state of commands.
2. Fix formatting, ID nits etc.

Changes from -03 to -04:

1. Allow retrieval of shared annotations for READ-ONLY mailbox.
2. Clarification of annotation loss on implicit removal of \Noselect
    mailboxes.
3. Now requires roll-back of all changes to matching mailboxes if
    there is a partial failure in SETANNOTATION.

Changes from -02 to -03:
1. Reworked entry naming scheme to split out mailbox name and use
    empty string for server items.

Changes from -01 to -02:
1. SETANNOTATION lists use (..).
2. Explicitly state behaviour of unsolicited responses.
3. Adding SHOULD behaviour for rename/delete of mailboxes.
4. Added statement about supporting annotations on \Noselect
    mailboxes.
5. Cleaned up formal syntax to use IMAP string type for entry and
    attributes, with requirements on how the string is formatted.
6. Use of ACAP vendor subtree registry for vendor tokens.

Changes from -00 to -01:
1. Multiple entry-att responses are now simply delimited by spaces
    in line with ANNOTATE spec. Adjusted examples to match.
2. Fixed entry-list formal syntax item to account for unsolicited
    parenthesised list of entries.
3. Removed setentries formal syntax item for simplicity since its
    only used once.
4. Removed reference to 'message annotation' in [section 5.1](#).
5. Changed formal syntax to restrict top level entries to /server
    and /mailbox/{...} only. Re-arranged entry names section to
    account for this change.
6. Added comment and example for ANNOTATION response to allow
    servers to return separate responses for each entry if desired.

Table of Contents

1. **Introduction and Overview**

   The ANNOTATEMORE extension is present in any IMAP4 [RFC3501]
   implementation which returns "ANNOTATEMORE" as one of the supported
   capabilities in the CAPABILITY command response.

   The goal of ANNOTATEMORE is to provide a means for clients to store
   and retrieve "metadata" on an IMAP4 server. The metadata can be
   associated with specific mailboxes or the server as a whole.

   The ANNOTATEMORE extension adds two new commands and one new untagged
   response to the IMAP4 base protocol.

   This extension makes the following changes to the IMAP4 protocol:

      adds a new SETANNOTATION command
      adds a new GETANNOTATION command
      adds a new ANNOTATION untagged response
      adds a new ANNOTATEMORE response code

   The data model used in ANNOTATEMORE is exactly the same as that used
   in the ANNOTATE [ANNOTATE] extension to IMAP4. This is modeled after
   that of the Application Configuration Access Protocol [RFC2244] with
   the exception of inheritance which is not deemed necessary here.

   The rest of this document describes the data model and protocol
   changes more rigorously.

2. **Data Model**

2.1 **Overview**

   The data model used in ANNOTATEMORE is one of a uniquely named entry
   with a set of uniquely named attributes, each of which has a value.
   Annotations can be added to mailboxes when a mailbox name is provided
   as the first argument to the new commands, or to the server as a
   whole when the empty string is provided as the first argument to the
   new commands.

   An annotation can contain multiple named entries. For example, a
   general comment being added to a mailbox may have an entry name of "/
   comment". This entry could include named attributes such as "value",
   "modifiedsince", "acl" etc to represent properties and data
   associated with the entry.

   The protocol changes to IMAP described below allow a client to access
   or change the values of any attributes in any entries in an
   annotation, assuming it has sufficient access rights to do so.

2.2 Namespace of entries and attributes

Each annotation is made up of a set of entries. Each entry has a hierarchical name in UTF-8, with each component of the name separated by a slash ("/").

Each entry is made up of a set of attributes. Each attribute has a hierarchical name in UTF-8, with each component of the name separated by a period (".").

The value of an attribute is NIL (has no value), or a string of zero or more octets.

Entry and attribute names are not permitted to contain asterisk ("*") or percent ("%") characters and MUST be valid UTF-8 strings which do not contain NUL. Invalid entry or attribute names result in a BAD response in any IMAP commands where they are used.

Use of non-visible UTF-8 characters in entry and attribute names is discouraged.

This specification defines an initial set of entry and attribute names available for use with mailbox and server annotations. In addition an extension mechanism is described to allow additional names to be added for extensibility.

2.2.1 Entry Names

Entry names MUST be specified in a standards track or IESG approved experimental RFC, or fall under the vendor namespace. See Section 5.1 for the registration template. This specification only allows two top-level entry types for servers and mailboxes.

2.2.1.1 Server Entries

These entries are used when the mailbox name argument to the new commands is the empty string.

/comment
   Defines a comment or note associated with the server.

/motd
   Defines a "message of the day" for the server. This entry is always read-only - clients cannot change it.

/admin
   Indicates a method for contacting the server administrator. This may be a URI (e.g. a mailto URL) or other contact information,

such as a telephone number. This entry is always read-only -
clients cannot change it.

/vendor/<vendor-token>
    Defines the top-level of entries associated with the server as
    created by a particular product of some vendor. This entry can be
    used by vendors to provide server or client specific attributes.
    The vendor-token MUST be registered with IANA, using the ACAP
    [RFC2244] vendor subtree registry.

2.2.1.2 Mailbox Entries

These entries are used when the mailbox name argument to the new
commands is not the empty string.

/comment
    Defines a comment or note associated with a mailbox.

/sort
    Defines the default sort criteria [SORT] to use when first
    displaying the mailbox contents to the user, or NIL if sorting is
    not required.

/thread
    Defines the default thread criteria [SORT] to use when first
    displaying the mailbox contents to the user, or NIL if threading
    is not required. If both sort and thread are not NIL, then
    threading should take precedence over sorting.

/check
    Boolean value "true" or "false" that indicates whether this
    mailbox should be checked at regular intervals by the client. The
    interval in minutes is specified by the /checkperiod entry.

/checkperiod
    Numeric value indicating a period of minutes to use for checking a
    mailbox that has the /check entry set to "true".

/vendor/<vendor-token>
    Defines the top-level of entries associated with a specific
    mailbox as created by a particular product of some vendor. This
    entry can be used by vendors to provide client specific
    attributes. The vendor-token MUST be registered with IANA, using
    the ACAP [RFC2244] vendor subtree registry.

2.2.2 Attribute Names

Attribute names MUST be specified in a standards track or IESG

approved experimental RFC, or fall under the vendor namespace. See
Section 5.1 for the registration template.

All attribute names implicitly have a ".priv" and a ".shared" suffix
which maps to private and shared versions of the entry. Searching or
fetching without using either suffix includes both. The client MUST
specify either a ".priv" or ".shared" suffix when storing an
annotation.

value
   The data value of the attribute.

size
   The size of the value, in octets. Set automatically by the server,
   read-only to clients.

modifiedsince
   An opaque value set by the server when this entry is modified. It
   can be used by the client to request notification of which entries
   have changed since a particular point in time and is useful for
   disconnected/synchronisation operations.

content-type
   A MIME [RFC2046] content type and subtype that describes the
   nature of the content of the "value" attribute.

vendor.<vendor-token>
   Defines an attribute associated with a particular product of some
   vendor. This attribute can be used by vendors to provide client
   specific attributes. The vendor-token MUST be registered with
   IANA, using the ACAP [RFC2244] vendor subtree registry.

2.3 Private versus Shared and Access Control

As discussed in the ANNOTATE [ANNOTATE] extension there is a need to
support both private and shared annotations. This document adopts the
scheme used in [ANNOTATE] that adds two standard suffixes for all
attributes: ".shared" and ".priv". A GETANNOTATION command which
specifies neither uses both. SETANNOTATION commands MUST explicitly
use .priv or .shared suffixes.

A user can only store and retrieve private annotations on a mailbox
which is returned to them via a LIST or LSUB command. A user can only
store and retrieve shared annotations on a mailbox that they can
SELECT and which opens READ-WRITE. A user can only retrieve shared
annotations on a mailbox that they can SELECT and which opens
READ-ONLY. If a client attempts to store a shared annotation on a
READ-ONLY mailbox, the server MUST respond with a NO response.

[3](). IMAP Protocol Changes

[3.1]() General Considerations

   The new commands and response each have a mailbox name argument,
   indicating that the annotations being referred to are attached to the
   specified mailbox. An empty string can be used for the mailbox name
   to signify server annotations.

   Both "*" and "%" list wildcard characters MAY be used in the mailbox
   name argument to commands to match all possible occurrences of a
   mailbox name pattern. However, "*" or "%" by themselves MUST NOT
   match the empty string (server) entries. Server entries can only be
   accessed by explicitly using the empty string as the mailbox name.

   Servers SHOULD ensure that mailbox annotations are automatically
   moved when the mailbox they refer to is renamed, i.e. the annotations
   follow the mailbox. Servers SHOULD delete annotations for a mailbox
   when the mailbox is deleted, so that a mailbox created with the same
   name as a previously existing mailbox does not inherit the old
   mailbox annotations. Servers SHOULD allow annotations on all 'types'
   of mailbox, including ones reporting \Noselect for their LIST
   response. Servers can implicitly remove \Noselect mailboxes when all
   child mailboxes are removed, and as such any annotations associated
   with the \Noselect mailbox SHOULD be removed.

   The server is allowed to impose limitations on the size of any one
   annotation or the total number of annotations for a single mailbox or
   for the server as a whole. However, the server MUST accept a minimum
   annotation data size of at least 1024 bytes, and a minimum annotation
   count per server or mailbox of at least 10.

[3.2]() GETANNOTATION Command

   This extension adds the GETANNOTATION command. This allows clients to
   retrieve annotations.

   This command is only available in authenticated or selected state
   [[RFC3501]()].

      Arguments: mailbox-name
                  entry-specifier
                  attribute-specifier

      Responses: required ANNOTATION response

      Result: OK - command completed
                  NO - command failure: can't access annotations on that
mailbox

                    BAD - command unknown or arguments invalid

   The mailbox-name argument MUST be a valid mailbox name or the empty
   string. In the later case, the annotations being referred to are the
   ones for the server as a whole.

   Example:

          C: a GETANNOTATION "" "/comment" "value.priv"
          S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
          S: a OK GETANNOTATION complete

      In the above example, the contents of the "value" attribute for
      the "/comment" server entry is requested by the client and
      returned by the server.

   "*" and "%" wildcard characters can be used in either specifier to
   match one or more characters at that position, with the exception
   that "%" does not match the hierarchy delimiter for the specifier it
   appears in (i.e. "/" for an entry specifier or "." for an attribute
   specifier). Thus an entry specifier of "/%" would match entries such
   as "/comment" and "/version", but not "/comment/note".

   Example:

          C: a GETANNOTATION "" "/*" "value.priv"
          S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
                          "/version" ("value.priv" "1.1")
                          "/motd/today" ("value.priv" "Closed at 1 pm")
          S: a OK GETANNOTATION complete

      In the above example, the contents of the "value" attributes for
      any server entries are requested by the client and returned by the
      server.

   Example:

          C: a GETANNOTATION "" "/%" "value"
          S: * ANNOTATION "/comment" ("value.priv" "My comment")
                                     ("value.shared" "Your comment")
                        "/motd" ("value.priv" "Its sunny outside!"
                                     ("value.shared" "Today is a Holiday")
          S: a OK GETANNOTATION complete

      In the above example, the contents of the "value" attributes for
      server entries at the top level of the entry hierarchy only, are
      requested by the client and returned by the server. Both the .priv
      and .shared values are returned, as neither were explicitly

requested.

Entry and attribute specifiers can be lists of atomic specifiers, so
that multiple items of each type may be returned in a single
GETANNOTATION command.

Example:

```
        C: a GETANNOTATION "" ("/comment" "/motd") "value.priv"
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
                           "/motd" ("value.priv" "Its sunny outside!")
        S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" attributes for
the two server entries "/comment" and "/motd" are requested by the
client and returned by the server.

Example:

```
        C: a GETANNOTATION "" "/comment" ("value.priv" "modifiedsince.priv")
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment"
                                       "modifiedsince.priv"
"19990203205432")
        S: a OK GETANNOTATION complete
```

In the above example, the contents of the "value" and
"modifiedsince" attributes for the "/comment" server entry are
requested by the client and returned by the server.

3.3 SETANNOTATION Command

This extension adds the SETANNOTATION command. This allows clients to
set annotations.

This command is only available in authenticated or selected state
[RFC3501].

```
   Arguments:  mailbox-name
               entry
               attribute-value
               list of entry, attribute-values

   Responses:  no specific responses for this command

   Result:     OK - command completed
               NO - command failure: can't set annotations,
                    or annotation too big or too many
               BAD - command unknown or arguments invalid
```

Sets the specified list of entries by adding or replacing the
specified attributes with the values provided. Clients can use NIL
for values of attributes it wants to remove from entries. The server
MAY return an unsolicited ANNOTATION response containing the updated
annotation data. The server SHOULD send such responses for any
changes to the server annotations, and SHOULD send such responses for
mailbox annotations only for the currently selected mailbox. Clients
MUST NOT assume that an unsolicited ANNOTATION response will be sent,
and MUST assume that if the command succeeds then the annotation has
been changed.

If the server is unable to store an annotation because the size of
its value is too large, the server MUST return a tagged NO response
with a "[ANNOTATEMORE TOOBIG]" response code.

If the server is unable to store a new annotation because the maximum
number of allowed annotations has already been reached, the server
MUST return a tagged NO response with a "[ANNOTATEMORE TOOMANY]"
response code.

If the server is unable to store the annotations for one or more
mailboxes matching the mailbox-name pattern, then the SETANNOTATION
command MUST fail and there MUST NOT be any changes to any of the
matching mailboxes, even those for which annotations could have been
changed successfully.

Example:

        C: a SETANNOTATION "INBOX" "/comment" ("value.priv" "My new
comment")
        S: a OK SETANNOTATION complete

    In the above example, the entry "/comment" for the mailbox "INBOX"
    is created (if not already present) and the private attribute
    "value" with data set to "My new comment" is created if not
    already present, or replaced if it previously exists.

Example:

        C: a SETANNOTATION "INBOX" "/comment" ("value.priv" NIL)
        S: a OK SETANNOTATION complete

    In the above example, the private "value" attribute of the entry
    "/comment" is removed from the mailbox "INBOX".

Multiple entries can be set in a single SETANNOTATION command by
listing entry-attribute-value pairs in the list.

Example:

```
        C: a SETANNOTATION "INBOX.%" "/comment" ("value.priv" "My new
comment")
        S: a OK SETANNOTATION complete
```

In the above example, the entry "/comment" for all mailboxes at
the top-level of the "INBOX" hierarchy are created (if not already
present) and the private and shared attributes "value" are created
respectively for each entry if not already present, or replaced if
they previously existed.

Multiple attributes can be set in a single SETANNOTATION command by
listing multiple attribute-value pairs in the entry list.

Example:

```
        C: a SETANNOTATION "INBOX" "/comment"
                           ("value.priv" "My new comment"
                            "vendor.foobar.bloop.priv" "foo's bar")
        S: a OK SETANNOTATION complete
```

In the above example, the entry "/comment" for the mailbox "INBOX"
is created (if not already present) and the private attributes
"value" and "vendor.foobar.bloop" are created if not already
present, or replaced if they previously existed.

Example:

```
        C: a SETANNOTATION "INBOX" "/comment" ("value.priv" "My new
comment")
        S: a NO [ANNOTATEMORE TOOMANY] SETANNOTATION failed
```

In the above example, the server is unable to store the requested
(new) annotation as it has reached the limit on the number of
annotations it can support on the specified mailbox.

3.4 ANNOTATION Response

The ANNOTATION response displays results of a GETANNOTATION command,
or can be returned as a unsolicited response at anytime by the server
in response to a change in an annotation.

Servers SHOULD send unsolicited ANNOTATION responses if changed by a
third-party, allowing servers to keep clients updated with changes to
annotations by other clients. Unsolicited mailbox entries MUST only
be returned for the currently selected mailbox. In this case, only
the entries are returned in the ANNOTATION response, not the
attributes and values. If the client wants to update any cached
values it must explicitly retrieve those using a GETANNOTATION
command.

Separate ANNOTATION responses MUST be used when more than one mailbox
matches the mailbox name argument pattern to the command.

The ANNOTATION response can contain multiple entries in a single
response, but the server is free to return multiple responses for
each entry or group of entries if it desires.

This response is only available in authenticated state [RFC3501].

3.4.1 ANNOTATION response with attributes and values

The response consists of a list of entries each of which has a list
of attribute-value pairs.

Example:

```
        C: a GETANNOTATION "" "/comment" "value.priv"
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
        S: a OK GETANNOTATION complete
```

In the above example, a single entry with a single attribute-value
pair is returned by the server.

Example:

```
        C: a GETANNOTATION "" ("/comment" "/motd") "value.priv"
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
                        "/motd" ("value.priv" "Its sunny outside!")
        S: a OK GETANNOTATION complete
```

In the above example, two entries each with a single
attribute-value pair is returned by the server.

Example:

```
        C: a GETANNOTATION "" ("/comment" "/motd") "value.priv"
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment")
        S: * ANNOTATION "" "/motd" ("value.priv" "Its sunny outside!")
        S: a OK GETANNOTATION complete
```

In the above example, the server returns two separate responses
for each of the two entries requested.

Example:

```
        C: a GETANNOTATION "" "/comment" ("value.priv" "modifiedsince.priv")
        S: * ANNOTATION "" "/comment" ("value.priv" "My comment"
                                       "modifiedsince.priv"
"19990203205432")
```

```
        S: a OK GETANNOTATION complete
```

In the above example, a single entry with two attribute-value
pairs is returned by the server.

   Example:

```
        C: a GETANNOTATION "INBOX.%" "/comment" "value.priv"
        S: * ANNOTATION "INBOX.1" "/comment" ("value.priv" "My comment for
1")
        S: * ANNOTATION "INBOX.2" "/comment" ("value.priv" "My comment for
2")
        S: * ANNOTATION "INBOX.3" "/comment" ("value.priv" "My comment for
3")
        S: a OK GETANNOTATION complete
```

In the above example, separate responses are returned for each
matching mailbox, each containing a single entry with a single
attribute-value pair.

3.4.2 Unsolicited ANNOTATION response without attributes and values

The response consists of a parenthesised list of entries, each of
which have changed on the server.

   Example:

```
        C: a NOOP
        S: * ANNOTATION "" ("/comment")
        S: a OK NOOP complete
```

In the above example, the server indicates that the "/comment"
server entry has been changed.

   Example:

```
        C: a NOOP
        S: * ANNOTATION "" ("/comment" "/motd")
        S: a OK NOOP complete
```

In the above example, the server indicates a change to two server
entries.

   Example:

```
        C: a NOOP
        S: * ANNOTATION "" ("/motd")
        S: * ANNOTATION "INBOX" ("/comment")
        S: a OK NOOP complete
```

In the above example, the server indicates a change to a server
entry, and to the an entry for the currently selected mailbox.

4. Formal Syntax

The following syntax specification uses the Augmented Backus-Naur
Form (ABNF) notation as specified in [RFC2234].

Non-terminals referenced but not defined below are as defined by
[RFC3501].

Except as noted otherwise, all alphabetic characters are case-
insensitive. The use of upper or lower case characters to define
token strings is for editorial clarity only. Implementations MUST
accept these strings in a case-insensitive fashion.

```
     annotate-data     = "ANNOTATION" SP mailbox SP entry-list
                           ; empty string for mailbox implies
                           ; server annotation.

     att-value         = attrib SP value

     attrib            = string
                           ; dot-separated attribute name
                           ; MUST NOT contain "*" or "%"
     attrib-match      = string
                           ; dot-separated attribute name
                           ; MAY contain "*" or "%" for use as wildcards

     attribs           = attrib-match / "(" attrib-match *(SP attrib-match)
")"
                           ; attribute specifiers that can include wildcards

     command-auth      /= setannotation / getannotation
                           ; adds to original IMAP4 command

     entries           = entry-match / "(" entry-match *(SP entry-match) ")"
                           ; entry specifiers that can include wildcards

     entry             = string
                           ; slash-separated path to entry
                           ; MUST NOT contain "*" or "%"

     entry-att         = entry SP "(" att-value *(SP att-value) ")"

     entry-list        = entry-att *(SP entry-att) /
                           "(" entry *(SP entry) ")"
                           ; entry attribute-value pairs list for
                           ; GETANNOTATION response, or
```

```
                            ; parenthesised entry list for unsolicited
                            ; notification of annotation changes

     entry-match        = string
                            ; slash-separated path to entry
                            ; MAY contain "*" or "%" for use as wildcards

     getannotation      = "GETANNOTATION" SP list-mailbox SP entries SP
attribs
                            ; empty string for list-mailbox implies
                            ; server annotation.

     response-data      /= "*" SP annotate-data CRLF
                            ; adds to original IMAP4 data responses

     resp-text-code     =/ "ANNOTATEMORE" SP "TOOBIG" /
                             "ANNOTATEMORE" SP "TOOMANY"
                            ; new response codes for SETANNOTATION failures

     setannotation      = "SETANNOTATION" SP list-mailbox SP setentryatt
                            ; empty string for list-mailbox implies
                            ; server annotation.

     setentryatt        = entry-att / "(" entry-att *(SP entry-att) ")"

     value              = nstring
```

5. IANA Considerations

   Both entry names and attribute names MUST be specified in a standards
   track or IESG approved experimental RFC, or fall under the vendor
   namespace.

[5.1](#) Entry and Attribute Registration Template

        To: iana@iana.org
        Subject: IMAP ANNOTATEMORE Registration

        Please register the following IMAP ANNOTATEMORE item:

        [] Entry   [] Attribute

        [] Mailbox   [] Server

        Name: _____

        Description: _____

        _____

        _____

        Contact person: _____

              email: _____

[6](#). Security Considerations

   The ANNOTATEMORE extension does not raise any security considerations
   that are not present in the base [[RFC3501](#)] protocol, and these issues
   are discussed in [[RFC3501](#)].

   Care must be taken to ensure that annotations whose values are
   intended to remain private are not stored in mailboxes which are
   accessible to other users. This includes mailboxes owned by the user
   by whose ACLs permit access by others as well as any shared
   mailboxes.

Normative References

   [RFC2046] Freed, N. and N. Borenstein, "Multipurpose Internet Mail
              Extensions (MIME) Part Two: Media Types", [RFC 2046](#),
              November 1996.

   [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
              Requirement Levels", [BCP 14](#), [RFC 2119](#), March 1997.

   [RFC2234] Crocker, D. and P. Overell, "Augmented BNF for Syntax
              Specifications: ABNF", [RFC 2234](#), November 1997.

   [RFC2244]   Newman, C. and J. Myers, "ACAP -- Application
               Configuration Access Protocol", RFC 2244, November 1997.

   [RFC3501]   Crispin, M., "INTERNET MESSAGE ACCESS PROTOCOL - VERSION
               4rev1", RFC 3501, March 2003.

   [SORT]   Crispin, M. and K. Murchison, "Internet Message Access
               Protocol - Sort and Thread Extension",
               draft-ietf-imapext-sort-15.txt, April 2004.

Informative References

   [ANNOTATE]
               Daboo, C. and R. Gellens, "IMAP ANNOTATE Extension",
               draft-ietf-imapext-annotate-09.txt, April 2004.

Author's Address

   Cyrus Daboo
   Cyrusoft International, Inc.
   5001 Baum Blvd.
   Suite 650
   Pittsburgh, PA 15213
   US

   EMail: daboo@cyrusoft.com

Appendix A. **Acknowledgments**

   The ideas expressed in this document are based on the message
   annotation document that was co-authored by Randall Gellens.

Intellectual Property Statement

   The IETF takes no position regarding the validity or scope of any
   intellectual property or other rights that might be claimed to
   pertain to the implementation or use of the technology described in
   this document or the extent to which any license under such rights
   might or might not be available; neither does it represent that it
   has made any effort to identify any such rights. Information on the
   IETF's procedures with respect to rights in standards-track and
   standards-related documentation can be found in BCP-11. Copies of
   claims of rights made available for publication and any assurances of
   licenses to be made available, or the result of an attempt made to
   obtain a general license or permission for the use of such
   proprietary rights by implementors or users of this specification can
   be obtained from the IETF Secretariat.

   The IETF invites any interested party to bring to its attention any
   copyrights, patents or patent applications, or other proprietary
   rights which may cover technology that may be required to practice
   this standard. Please address the information to the IETF Executive
   Director.

HEREIN WILL NOT INFRINGE ANY RIGHTS OR ANY IMPLIED WARRANTIES OF
MERCHANTABILITY OR FITNESS FOR A PARTICULAR PURPOSE.

Acknowledgment