                  Message Tracking Query Protocol

Status of this Memo

   This document specifies an Internet standards track protocol for the
   Internet community, and requests discussion and suggestions for
   improvements. Please refer to the current edition of the "Internet
   Official Protocol Standards" (STD 1) for the standardization state
   and status of this protocol. Distribution of this memo is unlimited.

Copyright Notice

Abstract

   Customers buying enterprise message systems often ask: Can I track
   the messages? Message tracking is the ability to find out the path
   that a particular message has taken through a messaging system and
   the current routing status of that message. This document describes
   the Message Tracking Query Protocol that is used in conjunction with
   extensions to the ESMTP protocol to provide a complete message
   tracking solution for the Internet.

1. Introduction

   The Message Tracking Models and Requirements document
   [RFC-MTRK-MODEL] discusses the models that message tracking solutions
   could follow, along with requirements for a message tracking solution
   that can be used with the Internet-wide message infrastructure. This
   memo and its companions, [RFC-MTRK-ESMTP] and [RFC-MTRK-TSN],
   describe a complete message tracking solution that satisfies those
   requirements. The memo [RFC-MTRK-ESMTP] defines an extension to the
   SMTP service that provides the information necessary to track
   messages. This memo defines a protocol that can be used to query the
   status of messages that have been transmitted on the Internet via
   SMTP. The memo [RFC-MTRK-TSN] describes the message/tracking-status
   [RFC-MIME] media type that is used to report tracking status
   information. Using the model document's terminology, this solution
   uses active enabling and active requests with both request and
   chaining referrals.

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 BCP 14, RFC 2119
   [RFC-KEYWORDS].

   All syntax descriptions use the ABNF specified by [RFC-ABNF].
   Terminal nodes not defined elsewhere in this document are defined in
   [RFC-ABNF], [RFC-URI], [RFC-MTRK-ESMTP], [RFC-SMTP], or
   [RFC-SMTPEXT].

2. Basic Operation

   The Message Tracking Query Protocol (MTQP) is similar to many other
   line-oriented Internet protocols, such as [POP3] and [NNTP].
   Initially, the server host starts the MTQP service by listening on
   TCP port 1038.

   When an MTQP client wishes to make use of the message tracking
   service, it establishes a TCP connection with the server host, as
   recorded from the initial message submission or as returned by a
   previous tracking request. To find the server host, the MTQP client
   first does an SRV lookup for the server host using DNS SRV records,
   with a service name of "mtqp" and a protocol name of "tcp", as in
   _mtqp._tcp.smtp3.example.com. (See the "Usage rules" section in
   [RFC-SRV] for details.) If the SRV records do not exist, the MTQP
   client then does an address record lookup for the server host. When
   the connection is established, the MTQP server sends a greeting. The
   MTQP client and MTQP server then exchange commands and responses
   (respectively) until the connection is closed or aborted.

2.1. Tracking Service DNS Considerations

   Because of the ways server host lookups are performed, many different
   tracking server host configurations are supported.

   A mail system that uses a single mail server host and has the MTQP
   server host on the same server host will most likely have a single MX
   record pointing at the server host, and if not, will have an address
   record. Both mail and MTQP clients will access that host directly.

   A mail system that uses a single mail server host, but wants tracking
   queries to be performed on a different machine, MUST have an SRV MTQP
   record pointing at that different machine.

A mail system that uses multihomed mail servers has two choices for
providing tracking services: either all mail servers must be running
tracking servers that are able to retrieve information on all
messages, or the tracking service must be performed on one (or more)
machine(s) that are able to retrieve information on all messages. In
the former case, no additional DNS records are needed beyond the MX
records already in place for the mail system. In the latter case,
SRV MTQP records are needed that point at the machine(s) that are
running the tracking service. In both cases, note that the tracking
service MUST be able to handle the queries for all messages accepted
by that mail system.

2.2. Commands

Commands in MTQP consist of a case-insensitive keyword, possibly
followed by one or more parameters. All commands are terminated by a
CRLF pair. Keywords and parameters consist of printable ASCII
characters. Keywords and parameters are separated by whitespace (one
or more space or tab characters). A command line is limited to 998
characters before the CRLF.

2.3. Responses

Responses in MTQP consist of a status indicator that indicates
success or failure. Successful commands may also be followed by
additional lines of data. All response lines are terminated by a
CRLF pair and are limited to 998 characters before the CRLF. There
are several status indicators: "+OK" indicates success; "+OK+"
indicates a success followed by additional lines of data, a multi-
line success response; "-TEMP" indicates a temporary failure; "-ERR"
indicates a permanent failure; and "-BAD" indicates a protocol error
(such as for unrecognized commands).

A status indicator MAY be followed by a series of machine-parsable,
case-insensitive response information giving more data about the
errors. These are separated from the status indicator and each other
by a single slash character ("/", decimal code 47). Following that,
there MAY be white space and a human-readable text message. The
human-readable text message is not intended to be presented to the
end user, but should be appropriate for putting in a log for use in
debugging problems.

In a multi-line success response, each subsequent line is terminated
by a CRLF pair and limited to 998 characters before the CRLF. When
all lines of the response have been sent, a final line is sent
consisting of a single period (".", decimal code 046) and a CRLF
pair. If any line of the multi-line response begins with a period,
the line is "dot-stuffed" by prepending the period with a second

period. When examining a multi-line response, the client checks to
see if the line begins with a period. If so, and octets other than
CRLF follow, the first octet of the line (the period) is stripped
away. If so, and if CRLF immediately follows the period, then the
response from the MTQP server is ended and the line containing the
".CRLF" is not considered part of the multi-line response.

An MTQP server MUST respond to an unrecognized, unimplemented, or
syntactically invalid command by responding with a negative -BAD
status indicator. A server MUST respond to a command issued when the
session is in an incorrect state by responding with a negative -ERR
status indicator.

2.4. Firewall Considerations

A firewall mail gateway has two choices when receiving a tracking
query for a host within its domain: it may return a response to the
query that says the message has been passed on, but no further
information is available; or it may perform a chaining operation
itself, gathering information on the message from the mail hosts
behind the firewall, and returning to the MTQP client the information
for each behind-the-firewall hop, or possibly just the final hop
information, possibly also disguising the names of any hosts behind
the firewall. Which option is picked is an administrative decision
and is not further mandated by this document.

If a server chooses to perform a chaining operation itself, it MUST
provide a response within 2 minutes, and SHOULD return a "no further
information is available" response if it cannot provide an answer at
the end of that time limit.

2.5. Optional Timers

An MTQP server MAY have an inactivity autologout timer. Such a timer
MUST be of at least 10 minutes in duration. The receipt of any
command from the client during that interval should suffice to reset
the autologout timer. An MTQP server MAY limit the number of
commands, unrecognized commands, or total connection time, or MAY use
other criteria, to prevent denial of service attacks.

An MTQP client MAY have an inactivity autologout timer while waiting
for a response from the server. Since an MTQP server may be a
firewall, and may be chaining information from other servers, such a
timer MUST be at least 2 minutes in duration.

3. Initialization and Option Response

   Once the TCP connection has been opened by an MTQP client, the MTQP
   server issues an initial status response that indicates its
   readiness. If the status response is positive (+OK or +OK+), the
   client may proceed with other commands.

   The initial status response MUST include the response information
   "/MTQP". Negative responses MUST include a reason code as response
   information. The following reason codes are defined here;
   unrecognized reason codes added in the future may be treated as
   equivalent to "unavailable".

      "/" "unavailable"
      "/" "admin"

   The reason code "/admin" SHOULD be used when the service is
   unavailable for administrative reasons. The reason code
   "/unavailable" SHOULD be used when the service is unavailable for
   other reasons.

   If the server has any options enabled, they are listed as the multi-
   line response of the initial status response, one per line. An
   option specification consists of an identifier, optionally followed
   by option-specific parameters. An option specification may be
   continued onto additional lines by starting the continuation lines
   with white space. The option identifier is case insensitive. Option
   identifiers beginning with the characters "vnd." are reserved for
   vendor use. (See below.)

   One option specification is defined here:

   STARTTLS [1*WSP "required"]

   This capability MUST be listed if the optional STARTTLS command is
   enabled on the MQTP server and one or more certificates have been
   properly installed.

   It has one optional parameter: the word "required" (The parameters
   for STARTTLS are case-insensitive). If the server requires that TLS
   be used for some of the domains the server handles, the server MUST
   specify the "required" parameter.

3.1. Examples

   Example #1 (no options):
   S: +OK/MTQP MTQP server ready

   Example #2 (service temporarily unavailable):
   S: -TEMP/MTQP/admin Service down for admin, call back later

   Example #3 (service permanently unavailable):
   S: -ERR/MTQP/unavailable Service down

   Example #4 (alternative for no options):
   S: +OK+/MTQP MTQP server ready
   S: .

   Example #5 (options available):
   S: +OK+/MTQP MTQP server ready
   S: starttls
   S: vnd.com.example.option2 with parameters private to example.com
   S: vnd.com.example.option3 with a very long
   S: list of parameters
   S: .

4. TRACK Command

   Syntax:

   track-command = "TRACK" 1*WSP unique-envid 1*WSP mtrk-secret CRLF
     mtrk-secret = base64

   Unique-envid is defined in [RFC-MTRK-ESMTP]. Mtrk-secret is the
   secret A described in [RFC-MTRK-ESMTP], encoded using base64.

   When the client issues the TRACK command, and the user is validated,
   the MTQP server retrieves tracking information about an email
   message. To validate the user, the value of mtrk-secret is hashed
   using SHA1, as described in [RFC-SHA1]. The hash value is then
   compared with the value passed with the message when it was
   originally sent. If the hash values match, the user is validated.

   A successful response MUST be multi-line, consisting of a [RFC-MIME]
   body part. The MIME body part MUST be of type multipart/related,
   with subparts of message/tracking-status, as defined in
   [RFC-MTRK-TSN]. The response contains the tracking information about
   the email message that used the given tracking-id. A negative
   response to the TRACK command may include these reason codes:

```
            "/" "tls-required"
            "/" "admin"
            "/" "unavailable"
            "/" "noinfo"
            "/" "insecure"
```

The reason code "/tls-required" SHOULD be used when the server has
decided to require TLS. The reason code "/admin" SHOULD be used when
the server has become unavailable, due to administrative reasons,
since the connection was initialized. The reason code "/unavailable"
SHOULD be used when the server has become unavailable, for other
reasons, since the connection was initialized. The reason code
"/insecure" is described later.

If a message has not been seen by the MTQP server, the server MUST
choose between two choices: it MAY return a positive response with an
action field of "opaque" in the tracking information, or it MAY
return a negative response with a reason code of "noinfo".

4.1. Examples

In each of the examples below, the unique-envid is
"<12345-20010101@example.com>", the secret A is "abcdefgh", and the
SHA1 hash B is (in hex) "734ba8b31975d0dbae4d6e249f4e8da270796c94".
The message came from example.com and the MTQP server is
example2.com.

```
Example #6      Message Delivered:
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon, 1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .
```

Example #7   Message Transferred:
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%; type=tracking-status
S:
S: --%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon, 1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: transferred
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S: Status:2.4.0
S:
S: --%%%--
S: .

Example #8 Message Delayed and a Dot-Stuffed Header:
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%; type=tracking-status
S: ..Dot-Stuffed-Header: as an example
S:
S: --%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon, 1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: delayed
S: Status: 4.4.1 (No answer from host)
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S: Will-Retry-Until: Thu, 4 Jan 2001 15:15:15 -0500
S:
S: --%%%--
S: .

Example #9 Two Users, One Relayed, One Failed:
```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user2@example1.com
S: Action: failed
S: Status 5.2.2 (Mailbox full)
S: Remote-MTA: dns; example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S:
S: --%%%%--
S: .
```

Example #10 Firewall:
```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; smtp.example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S:
```

```
S: --%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; smtp.example3.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example3.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%--
S: .
```

Example #11 Firewall, Combining Per-Recipient Blocks:
```
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%; type=tracking-status
S:
S: --%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon,  1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; smtp.example3.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example3.com
S: Action: delivered
S: Status:2.5.0
S:
S: --%%%--
S: .
```

Example #12 Firewall, Hiding System Names Behind the Firewall:
C: TRACK <12345-20010101@example.com> YWJjZGVmZ2gK
S: +OK+ Tracking information follows
S: Content-Type: multipart/related; boundary=%%%%; type=tracking-status
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon, 1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user1@example1.com
S: Final-Recipient: rfc822; user1@example1.com
S: Action: relayed
S: Status: 2.1.9
S: Remote-MTA: dns; example2.com
S: Last-Attempt-Date: Mon, 1 Jan 2001 19:15:03 -0500
S:
S: --%%%%
S: Content-Type: message/tracking-status
S:
S: Original-Envelope-Id: 12345-20010101@example.com
S: Reporting-MTA: dns; example2.com
S: Arrival-Date: Mon, 1 Jan 2001 15:15:15 -0500
S:
S: Original-Recipient: rfc822; user2@example1.com
S: Final-Recipient: rfc822; user4@example1.com
S: Action: delivered
S: Status: 2.5.0
S:
S: --%%%%--
S: .

5. COMMENT Command

    Syntax:

      comment-command = "COMMENT" opt-text CRLF
             opt-text = [WSP *(VCHAR / WSP)]

    When the client issues the COMMENT command, the MTQP server MUST
    respond with a successful response (+OK or +OK+). All optional text
    provided with the COMMENT command are ignored.

6. STARTTLS Command

    Syntax:

        starttls-command = "STARTTLS" 1*WSP domain *WSP CRLF
                    domain = (sub-domain 1*("." sub-domain))

    TLS [TLS] is a popular mechanism for enhancing TCP communications
    with confidentiality and authentication. All MTQP servers MUST
    implement TLS. However, TLS MAY be disabled by a server
    administrator, either explicitly or by failing to install any
    certificates for TLS to use. If an MTQP server supports TLS and has
    one or more certificates available it MUST include "STARTTLS" in the
    option specifications list on protocol startup.

        Note: TLS SHOULD be enabled on MQTP servers whenever possible.

    The parameter MUST be a fully qualified domain name (FQDN). A client
    MUST specify the hostname it believes it is speaking with so that the
    server may respond with the proper TLS certificate. This is useful
    for virtual servers that provide message tracking for multiple
    domains (i.e., virtual hosting).

    If the server returns a negative response, it MAY use one of the
    following response codes:
        "/" "unsupported"
        "/" "unavailable"
        "/" "tls-in-progress"
        "/" "bad-fqdn"

    If TLS is not supported, then a response code of "/unsupported"
    SHOULD be used. If TLS is not available for some other reason, then
    a response code of "/unavailable" SHOULD be used. If a TLS session
    is already in progress, then it is a protocol error and "-BAD" MUST
    be returned with a response code of "/tls-in-progress". If there is
    a mismatch between the supplied FQDN and the FQDN found in the
    dNSName field of the subjectAltName extension of the server's
    certificate [RFC-X509], then it is a protocol error and "-BAD" MUST
    be returned with a response code of "/bad-fqdn".

    After receiving a positive response to a STARTTLS command, the client
    MUST start the TLS negotiation before giving any other MTQP commands.

    If the MTQP client is using pipelining (see below), the STARTTLS
    command must be the last command in a group.

6.1. Processing After the STARTTLS Command

If the TLS handshake fails, the server SHOULD abort the connection.

After the TLS handshake has been completed, both parties MUST
immediately decide whether or not to continue based on the
authentication and confidentiality achieved. The MTQP client and
server may decide to move ahead even if the TLS negotiation ended
with no authentication and/or no confidentiality because most MTQP
services are performed with no authentication and no confidentiality,
but some MTQP clients or servers may want to continue only if a
particular level of authentication and/or confidentiality was
achieved.

If the MTQP client decides that the level of authentication or
confidentiality is not high enough for it to continue, it SHOULD
issue an MTQP QUIT command immediately after the TLS negotiation is
complete.

If the MTQP server decides that the level of authentication or
confidentiality is not high enough for it to continue, it MAY abort
the connection. If it decides that the level of authentication or
confidentiality is not high enough for it to continue, and it does
not abort the connection, it SHOULD reply to every MTQP command from
the client (other than a QUIT command) with a negative "-ERR"
response and a response code of "/insecure".

6.2. Result of the STARTTLS Command

Upon completion of the TLS handshake, the MTQP protocol is reset to
the initial state (the state in MTQP after a server starts up). The
server MUST discard any knowledge obtained from the client prior to
the TLS negotiation itself. The client MUST discard any knowledge
obtained from the server, such as the list of MTQP options, which was
not obtained from the TLS negotiation itself.

At the end of the TLS handshake, the server acts as if the connection
had been initiated and responds with an initial status response and,
optionally, a list of server options. The list of MTQP server
options received after the TLS handshake MUST be different than the
list returned before the TLS handshake. In particular, a server MUST
NOT return the STARTTLS option in the list of server options after a
TLS handshake has been completed.

Both the client and the server MUST know if there is a TLS session
active. A client MUST NOT attempt to start a TLS session if a TLS
session is already active.

7. QUIT Command

     Syntax:

        quit-command = "QUIT" CRLF

   When the client issues the QUIT command, the MTQP session terminates.
   The QUIT command has no parameters. The server MUST respond with a
   successful response. The client MAY close the session from its end
   immediately after issuing this command (if the client is on an
   operating system where this does not cause problems).

8. Pipelining

   The MTQP client may elect to transmit groups of MTQP commands in
   batches without waiting for a response to each individual command.
   The MTQP server MUST process the commands in the order received.

   Specific commands may place further constraints on pipelining. For
   example, STARTTLS must be the last command in a batch of MTQP
   commands.

8.1. Examples

   The following two examples are identical:

   Example #13 :
   C: TRACK <tracking-id> YWJjZGVmZ2gK
   S: +OK+ Tracking information follows
   S:
   S: ... tracking details #1   go here ...
   S: .
   C: TRACK <tracking-id-2> QUJDREVGR0gK
   S: +OK+ Tracking information follows
   S:
   S: ... tracking details #2   go here ...
   S: .

```
   Example #14 :
   C: TRACK <tracking-id> YWJjZGVmZ2gK
   C: TRACK <tracking-id-2> QUJDREVGR0gK
   S: +OK+ Tracking information follows
   S:
   S: ... tracking details #1      go here ...
   S: .
   S: +OK+ Tracking information follows
   S:
   S: ... tracking details #2      go here ...
   S: .
```

9. The MTQP URI Scheme

9.1. Intended usage

   The MTQP URI scheme is used to designate MTQP servers on Internet
   hosts accessible using the MTQP protocol. It performs an MTQP query
   and returns tracking status information.

9.2. URI Scheme Name

   The name of the URI scheme is "mtqp".

9.3. URI Scheme Syntax

   An MTQP URI takes one of the following forms:

```
   mtqp://<mserver>/track/<unique-envid>/<mtrk-secret>
   mtqp://<mserver>:<port>/track/<unique-envid>/<mtrk-secret>
```

   The first form is used to refer to an MTQP server on the standard
   port, while the second form specifies a non-standard port. Both of
   these forms specify that the TRACK command is to be issued using the
   given tracking id (unique-envid) and authorization secret (mtrk-
   secret). The path element "/track/" MUST BE treated case
   insensitively, but the unique-envid and mtrk-secret MUST NOT be.

9.3.1. Formal Syntax

   This is an ABNF description of the MTQP URI.

```
   mtqp-uri = "mtqp://" authority "/track/" unique-envid "/" mtrk-secret
```

9.4. Encoding Rules

The encoding of unique-envid is discussed in [RFC-MTRK-ESMTP].
Mtrk-secret is required to be base64 encoded. If the "/", "?" and
"%" octets appear in unique-envid or mtrk-secret, they are further
required to be represented by a "%" followed by two hexadecimal
characters. (The two characters give the hexadecimal representation
of that octet).

10. IANA Considerations

System port number 1038 has been assigned to the Message Tracking
Query Protocol by the Internet Assigned Numbers Authority (IANA).

The service name "MTQP" has been registered with the IANA.

The IANA has also registered the URI registration template found in
Appendix A in accordance with [BCP35].

This document requests that IANA maintain one new registry: MTQP
options. The registry's purpose is to register options to this
protocol. Options whose names do not begin with "vnd." MUST be
defined in a standards track or IESG approved experimental RFC. New
MTQP options MUST include the following information as part of their
definition:

    option identifier
    option parameters
    added commands
    standard commands affected
    specification reference
    discussion

One MTQP option is defined in this document, with the following
registration definition:

    option identifier: STARTTLS
    option parameters: none
    added commands: STARTTLS
    standard commands affected: none
    specification reference: RFC 3887
    discussion: see RFC 3887

Additional vendor-specific options for this protocol have names that
begin with "vnd.". After the "vnd." would appear the reversed domain
name of the vendor, another dot ".", and a name for the option
itself. For example, "vnd.com.example.extinfo" might represent a

vendor-specific extension providing extended information by the owner
of the "example.com" domain. These names MAY be registered with
IANA.

11. Security Considerations

If the originator of a message were to delegate his or her tracking
request to a third party, this would be vulnerable to snooping over
unencrypted sessions. The user can decide on a message-by-message
basis if this risk is acceptable.

The security of tracking information is dependent on the randomness
of the secret chosen for each message and the level of exposure of
that secret. If different secrets are used for each message, then
the maximum exposure from tracking any message will be that single
message for the time that the tracking information is kept on any
MTQP server. If this level of exposure is too much, TLS may be used
to reduce the exposure further.

It should be noted that message tracking is not an end-to-end
mechanism. Thus, if an MTQP client/server pair decide to use TLS
confidentiality, they are not securing tracking queries with any
prior or successive MTQP servers.

Both the MTQP client and server must check the result of the TLS
negotiation to see whether acceptable authentication or
confidentiality was achieved. Ignoring this step completely
invalidates using TLS for security. The decision about whether
acceptable authentication or confidentiality was achieved is made
locally, is implementation-dependent, and is beyond the scope of this
document.

The MTQP client and server should note carefully the result of the
TLS negotiation. If the negotiation results in no confidentiality,
or if it results in confidentiality using algorithms or key lengths
that are deemed not strong enough, or if the authentication is not
good enough for either party, the client may choose to end the MTQP
session with an immediate QUIT command, or the server may choose to
not accept any more MTQP commands.

A man-in-the-middle attack can be launched by deleting the "STARTTLS"
option response from the server. This would cause the client not to
try to start a TLS session. An MTQP client can protect against this
attack by recording the fact that a particular MTQP server offers TLS
during one session and generating an alarm if it does not appear in
an option response for a later session.

Similarly, the identity of the server as expressed in the server's certificate should be cached, and an alarm generated if they do not match in a later session.

If TLS is not used, a tracking request is vulnerable to replay attacks, such that a snoop can later replay the same handshake again to potentially gain more information about a message's status.

Before the TLS handshake has begun, any protocol interactions are performed in the clear and may be modified by an active attacker. For this reason, clients and servers MUST discard any knowledge obtained prior to the start of the TLS handshake upon completion of the TLS handshake.

If a client/server pair successfully performs a TLS handshake and the server does chaining referrals, then the server SHOULD attempt to negotiate TLS at the same (or better) security level at the next hop. In a hop-by-hop scenario, STARTTLS is a request for "best effort" security and should be treated as such.

SASL is not used because authentication is per message rather than per user.

12. Protocol Syntax

This is a collected ABNF description of the MTQP protocol.

```
mtqp-uri = "mtqp://" authority "/track/" unique-envid "/" mtrk-secret

conversation = command-response *(client-command command-response)

; client side
client-command = track-command / starttls-command / quit-command
/comment-command

track-command = "TRACK" 1*WSP unique-envid 1*WSP mtrk-secret CRLF
mtrk-secret = base64

starttls-command = "STARTTLS" 1*WSP domain *WSP CRLF
domain = (sub-domain 1*("." sub-domain))

quit-command = "QUIT" CRLF

comment-command = "COMMENT" opt-text CRLF
```

```
; server side
command-response = success-response / temp-response / error-response /
bad-response

temp-response = "-TEMP" response-info opt-text CRLF

opt-text = [WSP *(VCHAR / WSP)]

error-response = "-ERR" response-info opt-text CRLF

bad-response = "-BAD" response-info opt-text CRLF

success-response = single-line-success / multi-line-success

single-line-success = "+OK" response-info opt-text CRLF

multi-line-success = "+OK+" response-info opt-text CRLF
                                   *dataline dotcrlf

dataline = *998OCTET CRLF

dotcrlf = "." CRLF

NAMECHAR = ALPHA / DIGIT / "-" / "_"

response-info = *(       "/" ( "admin" / "unavailable" / "unsupported"
/ "tls-in-progress" / "insecure" / "tls-required" / 1*NAMECHAR ) )
```

13. Acknowledgements

The description of STARTTLS is based on [RFC-SMTP-TLS].

14. References

14.1. Normative References

[RFC-MIME]   Freed, N. and N. Borenstein, "Multipurpose
                     Internet Mail Extensions (MIME) Part One: Format
                     of Internet Message Bodies", RFC 2045, November
                     1996.

[RFC-ABNF]   Crocker, D., Ed. and P. Overell, "Augmented BNF
                     for Syntax Specifications: ABNF", RFC 2234,
                     November 1997.

[RFC-SRV]   Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS
                     RR for specifying the location of services (DNS
                     SRV)", RFC 2782, February 2000.

[RFC-SMTP]   Klensin, J., "Simple Mail Transfer Protocol", RFC
                     2821, April 2001.

[RFC-SMTPEXT]   Myers, J., "SMTP Service Extension for
                     Authentication", RFC 2554, March 1999.

[RFC-MTRK-ESMTP]   Allman, E. and T. Hansen, "SMTP Service Extension
                     for Message Tracking", RFC 3885, September 2004.

[RFC-MTRK-MODEL]   Hansen, T., "Message Tracking Models and
                     Requirements", RFC 3885, September 2004.

[RFC-MTRK-TSN]   Allman, E., "The Message/Tracking-Status MIME
                     Extension", RFC 3886, September 2004.

[RFC-URI]   Berners-Lee, T., Fielding, R. and L. Masinter,
                     "Uniform Resource Identifiers (URI): Generic
                     Syntax", RFC 2396, August 1998.

[TLS]   Dierks, T. and C. Allen, "The TLS Protocol Version
                     1.0", RFC 2246, January 1999.

14.2. Informational References

[BCP35]   Petke, R. and I. King, "Registration Procedures
                        for URL Scheme Names", BCP 35, RFC 2717, November
                        1999.

[RFC-SHA1]   Eastlake, D. and P. Jones, "US Secure Hash
                        Algorithm 1 (SHA1)", RFC 3174, September 2001.

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

[RFC-SMTP-TLS]   Hoffman, P., "SMTP Service Extension for Secure
                        SMTP over Transport Layer Security", RFC 3207,
                        February 2002.

[RFC-X509]   Housley, R., Polk, W., Ford, W. and D. Solo,
                        "Internet X.509 Public Key Infrastructure
                        Certificate and Certificate Revocation List (CRL)
                        Profile", RFC 3280, April 2002.

[POP3]   Myers, J. and M. Rose, "Post Office Protocol -
                        Version 3", STD 53, RFC 1939, May 1996.

[NNTP]   Kantor, B. and P. Lapsley, "Network News Transfer
                        Protocol", RFC 977, February 1986.

Appendix A. MTQP URI Registration Template

   Scheme name: **mtqp**

   Scheme syntax: see section 9.1

   Character encoding considerations: see section 9.4

   Intended usage: see section 9.3

   Applications and/or protocols which use this scheme: MTQP

   Interoperability considerations: as specified for MTQP

   Security considerations: see section 11.0

   Relevant publications: [RFC-MTRK-ESMTP], [RFC-MTRK-MODEL],
   [RFC-MTRK-TSN]

   Contact: MSGTRK Working Group

   Author/Change Controller: IESG

Author's Address

   Tony Hansen
   AT&T Laboratories
   Middletown, NJ 07748
   USA

   Phone: +1.732.420.8934
   EMail: tony+msgtrk@maillennium.att.com

Full Copyright Statement

Intellectual Property

Acknowledgement