MADMAN Working Group
INTERNET-DRAFT
draft-ernst-msgmib-00.txt

Bruce Ernst [bruce_ernst@lotus.com]
Lotus Development Corporation
Gordon Jones [gbjones@mitre.org]
MITRE Corporation
Jonathan Weintraub
Electronic Data Systems, Inc.

March 1998

Message Tracking MIB

Status of this Memo

Abstract

This document defines a message tracking Management Information Base (MIB) for electronic messaging management. Message tracking is the ability to find out, after the fact, the path that a particular message took through the messaging system and the current status of that message.

1. The SNMP Network Management Framework.

The major components of the SNMP Network Management framework are described in the documents listed below.

- o RFC 1902 [1] defines the Structure of Management Information (SMI), the mechanisms used for describing and naming objects for the purpose of management.
- o STD 17, RFC 1213 [2] defines MIB-II, the core set of managed objects (MO) for the Internet suite of protocols.
- o RFC 1905 [3] defines the protocol used for network access to managed objects.

The framework is adaptable/extensible by defining new MIBs to suit the requirements of specific applications/protocols/situations.

Managed objects are accessed via a virtual information store, the MIB. Objects in the MIB are defined using the subset of Abstract Syntax Notation One (ASN.1) defined in the SMI. In particular, each object type is named by an OBJECT IDENTIFIER, which is an administratively assigned name. The object type together with an object instance serves to uniquely identify a specific instantiation of the object. For human convenience, often a textual string, termed the descriptor, is used to refer to the object type.

2. Message Tracking

Message tracking refers, in its simplest form, to determining the path a message
has taken and it's current location. This capability is analogous to the service
provided by carriers of conventional paper mail - the ability to quickly locate
where a package is, and to determine whether or not the package has been
delivered to its destination. This document discusses a specific SNMP-based
per-message mechanism to query the messaging system to perform message tracking. For a detailed definition of message tracking, and the need for message
tracking, refer to [MODEL].

3. MIB Data to Support Message Tracking

This MIB contains the syntax for message tracking via SNMP. The attributes
defined are identical to those defined for message tracking via e-mail, except
for those attributes that are required purely to enable SNMP-specific semantics.

When making use of SNMP, it is expected that the following mode of operation
will be used. The entity acting in the manager role issues one or more "SET"
operations filling in a row in a request table. The variables set in the request
table together comprise a message tracking query. The entity acting in the agent
role reads the values set in the request table, queries the local message store
(or usage logs), and returns information regarding messages that satify the query
criteria in a response table. The entity acting as the manager then reads the
response table and notifies an administrator or performs additional queries as
required.

In the message tracking MIB that follows, there are three tables:

1. **the mtaInformationTable**
2. **the msgTrackRequestTable**
3. **the msgTrackResponseTable**

The mtaInformationTable contains one entry for each MTA that is monitored
by this agent. The table contains information about the MTA as a whole, such
as: the name of the MTA and the length of time that the MTA has been recording
information. This last item is important to know, since if the MTA indicates that
it only has information about messages that have passed through it in the last two
days, and the administrator wishes to track a message that was sent four days ago,
then there will be no need to attempt to track the message through this MTA.

The msgTrackRequestTable is used by a manager to specify a query to be made
of the agent. It is not required that the manager have any knowledge of a unique
message identifier, but knowledge of some information pertaining to the message
in question is recommended. Since the query information may be matched by a
number of potential matches, it is possible to find out information about several
different messages.

The msgTrackResponseTable is a parallel to the msgTrackRequestTable in that it
holds the response(s) to the queries posed in the msgTrackRequestTable. The
manager reads this table to retrieve information about the messages that match
the input query information. The two tables share a common index to facilitate
navigation between them.

```
MESSAGE-TRACKING-MIB DEFINITIONS ::= BEGIN

   IMPORTS
       OBJECT-TYPE, MODULE-IDENTITY, Counter32
   FROM SNMPv2-SMI
       DisplayString, TimeInterval, DateAndTime
   FROM SNMPv2-TC;

--   Note that in SNMPv1 the IMPORTS section should look like the
following:
--   IMPORTS
--       OBJECT-TYPE, enterprises, Counter, Gauge, TimeTicks
--              FROM RFC1155-SMI
--       DisplayString
--             FROM RFC1213-MIB;


-- OBJECT IDENTIFIERS
  MsgTracking   OBJECT IDENTIFIER ::= { experimental 73 2 }

-- MODULE IDENTIFICATION   use in V2 version only
mta-message-track MODULE-IDENTITY
       LAST-UPDATED "9704110000Z"
       ORGANIZATION "IETF"
       CONTACT-INFO
             "Gordon  Jones
               Postal: 1820 Dolly Madison Boulevard
                           Mc Lean, VA 22102-3481
               Tel:    +1 703 883 7670
               Fax:    +1 703 883 7670
               E-mail: gbjones@mitre.org"
       DESCRIPTION
         "The MIB module describing message tracking"
       ::= { MsgTracking 1 }

-- Note that the MODULE-IDENTITY macro does not exist in SNMP version 1,
-- for a V1 implementation, replace the above macro with the following
line :
--  mta-message-track OBJECT IDENTIFIER ::= {MsgTracking 1 }


-- Note that the textual conventions DateAndTime and TimeInterval
-- do not exist in SNMPv1, for a V1 implementation include the following
-- 2 lines:
--   DateAndTime ::= DisplayString
--   TimeInterval ::= INTEGER (0..2147483647)
```

```
-- Define the NameForm textual convention for originator and recipient
-- names in this MIB
NameForm ::= INTEGER {
   freeForm(1),
   x400(2),
   smtp(3)
 }

-- Define the DispositionStatus textual convention which indicates the
-- disposition of a message by the MTA for a particular recipient.
Values
-- are:
-- unknown   - The disposition of the message could not be determined
-- transferred - The message was forwarded to another MTA for
-- delivery without name or content transformation.
-- delivered - The message was delivered to the intended recipient.
-- non-delivered - The message could not be delivered to the intended
-- recipient.
-- redirected - The message was forwarded to another MTA for
-- delivery.  The recipient name and/or messageId
-- may have changed as a result of this process.
-- dlist-expanded - The intended recipient was a distribution list
-- which was expanded by the MTA.
-- in-queue - The message for this recipient is currently being
-- processed by the MTA.

DispositionStatus ::= INTEGER {
   unknown(1),
   transferred(2),
   delivered(3),
   non-delivered(4),
   redirected(5),
   dlist-expanded(6),
   in-queue(7)
   }

-- Define the MsgType textual convention. Values are:
--
-- data
-- status
-- probe

MsgType ::= INTEGER {
  any(1),
  data(2),
  status(3),
  probe(4)
  }
```

```
-- How this mib works :
-- Most message tracking MIBs operate on the presupposition that the
manager
-- entering a query knows the unique message ID for the message being
tracked.
-- In practice, this is often not the case.  This MIB allows for a
2-step
-- query process - find the appropriate message, then track it.


--     +===============+
--     | Manager       | = 1 => +====================================+
--     +===============+ = 6 => | msgTrackRequestTable               |
--           ^                  +====================================+
--           |                           |
--           |                           2    +==========+
--           |                           + => | Agent    |
--           |                                +==========+
--           5                                     |
+======================+
--           |                                     + 3 => | Message
Store        |
--           |                             += 4 ==========
+======================+
--           |                             |
--           |                             v
--           |                 +=======================================+
--           ================> | msgTrackResponseTable                 |
--                             +=======================================+
--
--  STEP 1:
-- Using the index obtained from 'msgTrackNextRequestIndex', a manager
-- creates a conceptual row in the 'msgTrackRequestTable', filling in
-- as many message attributes as are known.  The manager also specifies
-- in 'reqMaxResponses', the maximim number of possible 'hits' for an
-- underspecified query.
--  STEP 2:
-- When the agent detects a new conceptual row in
'msgTrackRequestTable',
-- it forms a query to be executed against the message store(s).
--  STEP 3:
-- The agent issues the query against the message store and receives
some
-- response(s).
--  STEP 4:
-- The agent then transfers up to 'reqMaxResponses' possible responses
-- to newly created conceptual rows in the 'msgTrackResponseTable'. The
agent
```

```
-- then sets the value of 'reqResponseStatus' in the
'msgTrackRequestTable'
-- to notify the manager of the results of the query.
--  STEP 5:
-- The manager, having detected a change in 'reqResponseStatus', knows
that
-- the query is complete.  It reads the potential responses from the
-- 'msgTrackResponseTable' and presents them to the end-user.
```

```
--  STEP 6:
-- The manager then instructs the agent to destroy the conceptual row
-- created in the 'msgTrackRequestTable', which causes the agent to
-- destroy the corresponding entries in the 'msgTrackResponseTable'.
--  STEP 7:  (optional, not illustrated)
-- If the original query did not produce an adequate response, a new
entry
-- is created in the 'msgTrackRequestTable'.  Entries in this table are
-- never reused!
--
--

-- MESSAGE TRACKING REQUEST TABLE:

mtaInformationTable OBJECT-TYPE
 SYNTAX SEQUENCE OF mtaInformationEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
   "The table holding information about the MTA being queried.  A table
     is used because there may be multiple MTAs at a single host."
       ::= { mta-message-track 1 }


mtaInformationEntry OBJECT-TYPE
 SYNTAX  mtaInformationEntry
 MAX-ACCESS not-accessible
 STATUS current
 DESCRIPTION
   "One entry in the table holding information about the MTA being
     queried"
 INDEX { mtaIndex }
       ::= { mtaInformationTable 1 }

mtaInformationEntry ::=  SEQUENCE {
 mtaIndex
     INTEGER,
 mtaName
    DisplayString,
 mtaMessagingType
    DisplayString,
 startTimeforRecordedInformation
    DateAndTime,
 alternativeAgent
    DisplayString
}
```

```
mtaIndex   OBJECT-TYPE
 SYNTAX      INTEGER (1..2147483647)
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
   "The integer index into this table."
 ::= { mtaInformationEntry 1 }

mtaName  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
  "The name of the MTA described in this row of the table."
 ::= { mtaInformationEntry 2 }

mtaMessagingType  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
   " Common name of the messaging system of this MTA (e.g. X.400,
SMTP)."
 ::= { mtaInformationEntry 3 }

mtaStartTimeforRecordedInformation  OBJECT-TYPE
 SYNTAX      DateAndTime
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
   " The date/time of the oldest message tracking information
      available from this MTA."
 ::= { mtaInformationEntry 4 }

mtaAlternativeAgent  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
   "The name (or address) of another agent  that may have message
                  tracking information concerning this MTA."
 ::= { mtaInformationEntry 5 }
```

```
msgTrackNextRequestIndex  OBJECT-TYPE
 SYNTAX      Counter32      -- Note that in SNMPv1, the syntax is simply
'Counter'
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The index that may be used by a manager (requestor) on a
'set-request' PDU
    to create a new conceptual row in the msgTrackRequestTable table
(and
    thereby issue a message tracking query)."
 ::= { mta-message-track 2 }

msgTrackRequestTable OBJECT-TYPE
 SYNTAX     SEQUENCE OF MsgTrackRequestEntry
 ACCESS    not-accessible
 STATUS     current
 DESCRIPTION
   "The table holding all active message tracking requests."
 ::= { mta-message-track 3 }

msgTrackRequestEntry OBJECT-TYPE
 SYNTAX      MsgTrackRequestEntry
 ACCESS     not-accessible
 STATUS      current
 DESCRIPTION
   "The entry associated with each request for message information."
 INDEX { reqEntryIndex }
 ::= { msgTrackRequestTable 1 }

MsgTrackRequestEntry ::= SEQUENCE {
 reqEntryIndex
     INTEGER,
 reqRowStatus
     INTEGER,
 reqResponseStatus
     INTEGER,
 reqMaxResponses
     INTEGER,
 reqUniqueMsgId
     DisplayString,
 reqInboundMsgId
     DisplayString,
 reqOutboundMsgId
     DisplayString,
 reqInboundOriginator
     DisplayString,
```

```
reqOutboundOriginator
     DisplayString,
 reqOriginatorNameForm
     NameForm,
 reqInboundRecipient
     DisplayString,
 reqOutboundRecipient
     DisplayString,
 reqRecipientNameForm
     NameForm,
 reqSubject
     DisplayString,
reqMinMsgSize
     INTEGER,
 reqMaxMsgSize
     INTEGER,
 reqEarliestArrivalTime
     DateAndTime,
 reqLatestArrivalTime
     DateAndTime,
 reqDispositionStatus
     DispositionStatus,
 reqFailureReason
     DisplayString,
 reqMsgType
     MsgType,
 reqCollapseRecipients
     INTEGER
}

reqEntryIndex  OBJECT-TYPE
 SYNTAX     INTEGER (1..2147483647)
 ACCESS    read-only
 STATUS     current
 DESCRIPTION
   "The integer index into the msgTrackRequestTable table."
 ::= { msgTrackRequestEntry 1 }

reqRowStatus  OBJECT-TYPE
 SYNTAX     INTEGER {
   active(1),
   notInService(2),
   notReady(3),
```

```
   createAndGo(4),
   createAndWait(5),
   destroy(6)
        }
 ACCESS    read-write
 STATUS     current
 DESCRIPTION
   "The status of the conceptual row. These are mapped to the same
values as
     the RowStatus textual conversion in SNMPv2 and carry the same
semantics
     with one exception: the exception is that when a manager
(requestor) sets the
     value to destroy(6), this also has the added semantics of deleting
all
     conceptual rows in the msgTrackResponseTable table whose
respEntryIndex
     matches the reqEntryIndex of this conceptual row."
 ::= { msgTrackRequestEntry 2 }

reqResponseStatus  OBJECT-TYPE
 SYNTAX       INTEGER {
   unknown(1),
   inProgress(2),
   failedNoMatches(3),
   failedInvalidQuery(4),
   failedError(5),
   successUnderqualified(6),
   success(7)
         }
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Indicates the status of this query and its responses in the
    msgTrackResponseTable. Values are:
    unknown  -  The status of this query is not known.
    inProgress  -  The agent(responder) is still processing the request.
    failedNoMatches    -  The query has been processed and has produced
no
    matches.
    failedInvalidQuery  -  The query could not be processed  due to
invalid or
    missing data in the original query.
    FailedError  -  The query could not be processed due to an error in
the
    agent(responder).
    successUnderqualified  -  The query was successfully processed, but
the query
```

```
    was found to be underqualified.  That is, more reponses were found
than were
    specified in reqMaxResponses. reqMaxResponses entries were returned
in the
    msgTrackResponseTable.
    success  -  The query succeeded, returning from 1 to reqMaxResponse
    entries in the msgTrackResponseTable."
 ::= { msgTrackRequestEntry 3 }
```

```
reqMaxResponses  OBJECT-TYPE
 SYNTAX       INTEGER (1..100)
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Specifies the largest number of responses to be returned in the
    msgTrackResponseTable on an underspecified query (i.e. the
    maximum value of respMsgIndex in the msgTrackResponseTable
    conceptual row whose  respEntryIndex matches the reqEntryIndex of
this
    conceptual row)."
 ::= { msgTrackRequestEntry 4 }

reqUniqueMsgId  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Specifies a unique message id used internally by the MTA for
identification
    of a message.  This form of the message id may or may not be
identical to the
    inbound and/or outbound forms of the message id.  If specified,
this may be
    the only search criteria required. If the entire unique message id
is not
    specified, prefix matching is assumed. Set to an empty (zero
length) string if
    unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 5 }

reqInboundMsgId  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Specifies a unique message id as received by the MTA for
identification of
    a message.  This form of the message id may or may not be identical
to the
    internal and/or outbound forms of the message id.  If specified,
this may be the
    only search criteria required.  If the entire inbound message id is
not specified,
    prefix matching is assumed.  Set to an empty (zero length) string
if unknown or
    irrelevant to query."
```

```
 ::= { msgTrackRequestEntry 6 }

reqOutboundMsgId  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
```

```
   "Specifies a unique message id as transmitted by the MTA for
identification
    of a message.  This form of the message id may or may not be
identical to the
    internal and/or inbound forms of the message id.  If specified,
this may be the
    only search criteria required.  If the entire outbound message id
is not
    specified, prefix matching is assumed.  Set to an empty (zero
length) string if
    unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 7 }

reqInboundOriginator  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
"Identifies the originator of the message in its received form,
expressed in
    string format.  The style and format of this identifier varies
according to a
    specific messaging technology. As a result of potentially disparate
messaging
    technologies, this identifier is only guaranteed to be the name
known to the
    end-user on the first  MTA in the delivery sequence.  If
    reqOriginatorNameForm is set to 'x.400(2)' or 'smtp(3)', the
supplied
    attributes will be considered in the match.  Any attributes not
supplied will be
    wildcarded.  If reqOriginatorNameForm is set to 'freeForm(1)', this
value is
    assumed to be a substring of the originator name. Set to an empty
(zero
    length) string if unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 8 }

reqOutboundOriginator  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Identifies the originator of the message in its transmitted form,
expressed
    in string format.  The style and format of this identifier varies
according to a
```

```
    specific messaging technology. As a result of potentially disparate
messaging
    technologies this identifier is only guaranteed to be the name
known to the
    end-user on the last  MTA in the delivery sequence.   If
    reqOriginatorNameForm is set to 'x.400(2)' or 'smtp(3)', the
supplied attributes
    will be considered in the match.  Any attributes not supplied will
be
    wildcarded. If reqOriginatorNameForm is set to 'freeForm(1)', this
value is
    assumed to be a substring of the originator name.  Set to an empty
(zero
    length) string if unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 9 }

reqOriginatorNameForm  OBJECT-TYPE
 SYNTAX      NameForm
```

```
ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Identifies the name form of originator strings supplied in the
    reqInboundOriginator and/or reqOutboundOriginator values.  This
value is
    used by the agent to perform name form dependant parsing of these
values.
    If neither of these strings are supplied, this name form value is
irrelevant to
    the query.  A value of 'any(1)' implies that  no special parsing
should be
    performed on the originator names supplied."
 ::= { msgTrackRequestEntry 10 }

reqInboundRecipient  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Identifies one of the recipients (the one to be tracked) of the
message in
    its received form, expressed in string format.  The style and
format of this
    identifier varies according to a specific messaging technology.  As
a result
    of potentially disparate messaging technologies, this identifier is
only
    guaranteed to be the name an end-user  knows the recipient by on
the first
    MTA in the delivery sequence. If reqRecipientNameForm is set to
'x.400(2)'
    or 'smtp(3)', the supplied attributes will be considered in the
match.  Any
    attributes not supplied will be wildcarded. If reqRecipientNameForm
is set to
    'freeForm(1)', this value is assumed to be a substring of the
recipient name.
    Set to an empty (zero length) string if unknown or irrelevant to
query."
 ::= { msgTrackRequestEntry 11 }

reqOutboundRecipient  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
```

```
   "Identifies one of the recipients (the one to be tracked) of the
message in
    its transmitted form, expressed in string format.  The style and
format of this
    identifier varies according to a specific messaging technology.  As
a result of
    potentially disparate messaging technologies, this identifier is
only guaranteed
    to be the name an end-user knows the recipient by on the last MTA
in the
    delivery sequence. If reqRecipientNameForm is set to 'x.400(2)' or
'smtp(3)',
    the supplied attributes will be considered in the match.  Any
attributes not
    supplied will be wildcarded. If reqRecipientNameForm is set to
'freeForm(1)',
    this value is assumed to be a substring of the recipient name.  Set
to an empty
    (zero length) string if unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 12 }
```

```
reqRecipientNameForm  OBJECT-TYPE
 SYNTAX      NameForm
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Identifies the name form of recipient strings supplied in the
    reqInboundRecipient and/or reqOutboundRecipient values.  This value
is
    used by the agent to perform  name form dependant parsing of these
values.
    If neither of these strings are supplied, this name form value is
irrelevant to
    the query.  A value of 'any(1)' implies that no special parsing
should be
    performed on the recipient names supplied."
 ::= { msgTrackRequestEntry 13 }

reqSubject  OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Identifies a substring of the text of the 'Subject' attribute of
the message.
    Since some messaging technologies make it difficult for an MTA to
preserve
    this data, it may not be supported by all agents. Set to an empty
(zero length)
    string if unknown or irrelevant to query."
 ::= { msgTrackRequestEntry 14 }

reqMinMsgSize  OBJECT-TYPE
 SYNTAX      INTEGER (1..2147483647)
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "Specifies the minimum size of a message to be tracked (content,
excluding
    envelope), expressed in kilo-octets. Set both reqMinMsgSize and
    reqMaxMsgSize to zero if message size is irrelevant to the query."
 ::= { msgTrackRequestEntry 15 }

reqMaxMsgSize  OBJECT-TYPE
 SYNTAX      INTEGER (1..2147483647)
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
```

```
   "Specifies the maximum size of a message to be tracked (content,
excluding
   envelope), expressed in kilo-octets. Set both reqMinMsgSize and
   reqMaxMsgSize to zero if message size is irrelevant to the query."
 ::= { msgTrackRequestEntry 16 }
```

```
reqEarliestArrivalTime  OBJECT-TYPE
 SYNTAX      DateAndTime
 ACCESS      read-write
 STATUS      current
 DESCRIPTION
   "Specifies the earliest arrival time, at this MTA, for a message to
be
    tracked. Set to an empty (zero length) string if unknown or
irrelevant to
    query."
 ::= { msgTrackRequestEntry 17 }

reqLatestArrivalTime  OBJECT-TYPE
 SYNTAX      DateAndTime
 ACCESS      read-write
 STATUS      current
 DESCRIPTION
   "Specifies the latest arrival time, at this MTA, for a message to be
    tracked. Set to an empty (zero length) string if unknown or
irrelevant to
    query."
 ::= { msgTrackRequestEntry 18 }

reqDispositionStatus  OBJECT-TYPE
 SYNTAX     DispositionStatus
 ACCESS     read-write
 STATUS     current
 DESCRIPTION
   "Specifies the disposition status of the message for a particular
    recipient. Set to 'unknown(1)' if unknown or irrelevant to the
query."
 ::= {msgTrackRequestEntry 19 }

reqMsgType OBJECT-TYPE
 SYNTAX    MsgType
 ACCESS     read-write
 STATUS      current
 DESCRIPTION
   "The type of message to be tracked.
    Set to 'any(1)' if message type is unknown or irrelevant to the
query."
 ::= {msgTrackRequestEntry 20 }

reqCollapseRecipients  OBJECT-TYPE
 SYNTAX       INTEGER {
                        false(1),
                        true(2)
```

```
                        }
 ACCESS     read-write
 STATUS      current
```

```
 DESCRIPTION
   "If a value of 'true(2)' is specified, a single msgTrackResponseEntry
    will be created for each matching message regardless of the
    number of recipients.  If not specified or set to 'false(1)',
    a msgTrackResponseEntry will be created for each matching message
    and/or recipient.  This variable may be used in the case of a
    distribution list or a message with a large number of
    recipients."
 ::= { msgTrackRequestEntry 21 }

reqFailureReason OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "A textual description of why a message tracking request failed.
    This variable may be set by an agent when the reqResponseStatus
    is set to either 'failedInvalidQuery(4)', or 'failedError(5)'."
 ::= {msgTrackRequestEntry 22 }

-- MESSAGE RESPONSE TABLE ( QUERY RESULTS )

msgTrackResponseTable OBJECT-TYPE
 SYNTAX     SEQUENCE OF MsgTrackResponseEntry
 ACCESS    not-accessible
 STATUS     current
 DESCRIPTION
   "The table holding the response to all active message tracking
requests."
 ::= { mta-message-track 4 }

msgTrackResponseEntry OBJECT-TYPE
 SYNTAX      MsgTrackResponseEntry
 ACCESS     not-accessible
 STATUS      current
 DESCRIPTION
   "The entry associated with each response to a request for message
     information."
 INDEX { respEntryIndex, respMsgIndex }
 ::= { msgTrackResponseTable 1 }

MsgTrackResponseEntry ::= SEQUENCE {
 respEntryIndex
     INTEGER,
 respMsgIndex
     INTEGER,
```

```
respDispositionStatus
     DispositionStatus,
 respDispositionTime
     INTEGER,
 respNextHopMta
     DisplayString,
 respPrevHopMta
     DisplayString,
 respNonDeliveryReason
     DisplayString,
 respMsgArrivalTime
    DateAndTime,
 respMsgSize
     INTEGER,
 respMsgPriority
     DisplayString,
 respUniqueMsgId
     DisplayString,
 respInboundMsgId
     DisplayString,
respOutboundMsgId
     DisplayString,
 respInboundOriginator
     DisplayString,
 respOutboundOriginator
     DisplayString,
 respInboundRecipient
     DisplayString,
 respOutboundRecipient
     DisplayString,
 respSupplementalInformation
     DisplayString,
 respSubject
     DisplayString,
 respMsgType
     MsgType
}

respEntryIndex  OBJECT-TYPE
 SYNTAX     INTEGER (1..2147483647)
 ACCESS    read-only
 STATUS     current
 DESCRIPTION
   "The primary integer index into the msgTrackResponseTable table. It
matches
     the value of reqEntryIndex for the original request. "
 ::= { msgTrackResponseEntry 1 }
```

```
respMsgIndex OBJECT-TYPE
 SYNTAX      INTEGER (1..2147483647)
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The secondary integer index into the msgTrackResponseTable table.
For each
    value of respEntryIndex in the table, there may be multiple
conceptual rows
    indexed by respMsgIndex, each denoting a possible response to the
tracking
    query.  The maximum number of entries should have an upper bound of
the
    value of reqMaxResponses in the conceptual row of
msgTrackRequestTable
    that represents the original query request. "
 ::= { msgTrackResponseEntry 2 }

respDispositionStatus OBJECT-TYPE
 SYNTAX      DispositionStatus
 ACCESS      read-only
 STATUS      current
 DESCRIPTION
   "Indicates the disposition of this message by this MTA for this
recipient."
 ::= { msgTrackResponseEntry 3 }

rspDispositionTime OBJECT-TYPE
 SYNTAX      DateAndTime
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Time at which this MTA disposed of this message for this recipient."
 ::= { msgTrackResponseEntry 4 }

respNextHopMta OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Name of the MTA to which this message was sent. MADMAN-compliant
    MTA's should be addressed in the form '(<host-id>::<mtaName>)'."
 ::= { msgTrackResponseEntry 5 }

respPrevHopMta OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
```

```
STATUS      current
DESCRIPTION
```

```
   "Name of the MTA from which this message was received. MADMAN-
    compliant MTA's should be addressed in the form
    '(<host-id>::<mtaName>)'."
 ::= { msgTrackResponseEntry 6 }

respNonDeliveryReason OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "A textual representation representing the reason for non-delivery to
    this recipient.  No attempt is made to normalize these non-delivered
    reasons across systems, since this indicates a terminal condition."
 ::= { msgTrackResponseEntry 7 }

respMsgArrivalTime OBJECT-TYPE
 SYNTAX      DateAndTime
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Represents the time at which this message for this recipient
arrived at
    this MTA."
 ::= { msgTrackResponseEntry 8 }

respMsgSize OBJECT-TYPE
 SYNTAX      INTEGER(1..2147483647)
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Size of the message in kilo-octets."
 ::= { msgTrackResponseEntry 9 }

respMsgPriority OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Textual representation of the priority of the message.  No attempt
is
    made to normalize these values across disparate messaging
technologies."
 ::= { msgTrackResponseEntry 10 }

respUniqueMsgId OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
```

```
 STATUS      current
```

```
DESCRIPTION
   "The unique message identifier that the MTA assigned internally
    to the message."
 ::= { msgTrackResponseEntry 11 }

respInboundMsgId OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The unique message identifier that the 'previous hop' MTA assigned
    to the message.  If the 'previous' MTA uses a different messaging
technology
    or identifier scheme, this identifier serves to correlate the
message from
    MTA to MTA. If the 'previous' MTA uses the same technology, this
value
    is generally superfluous.  If this is the first MTA in the delivery
    sequence, or if the previous message id is unknown, this variable
is null-
    valued."
 ::= { msgTrackResponseEntry 12 }

respOutboundMsgId OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The unique message identifier that the 'next hop' MTA assigned
    to the message.  If the 'next' MTA uses a different messaging
technology
    or identifier scheme, this identifier serves to correlate the
message from
    MTA to MTA. If the 'next' MTA uses the same technology, this value
    is generally superfluous.  If this is the last MTA in the delivery
sequence,
    or if the next hop message id is unknown, this variable is
null-valued."
 ::= { msgTrackResponseEntry 13 }

respInboundOriginator OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Textual representation identifying the originator of the message as
it was
```

```
    received from the 'previous hop' MTA. The style of this variable
varies
    according to a specific messaging technology."
 ::= { msgTrackResponseEntry 14 }

respOutboundOriginator OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
```

```
DESCRIPTION
   "Textual representation identifying the originator of the message as
it
    was (or will be) presented to the 'next hop' MTA. The style of this
    variable varies according to a specific messaging technology."
 ::= { msgTrackResponseEntry 15 }

respInboundRecipient OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Textual representation identifying the recipient of the message as
it was
    received from the 'previous hop' MTA. The style of this variable
varies
    according to a specific messaging technology.."
 ::= { msgTrackResponseEntry 16 }

respOutboundRecipient OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Textual representation identifying the recipient of the message as
it
    was (or will be) presented to the 'next hop' MTA. The style of this
    variable varies according to a specific messaging technology."
 ::= { msgTrackResponseEntry 17 }

respSupplementalInformation OBJECT-TYPE
 SYNTAX      DisplayString
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "Contains information provided by the agent to the manager that may
be
    of use in identifying or tracking this message. No formal structure
for
    this information is specified.  Knowledge of the contents of this
field
    is by bilateral agreement."
 ::= { msgTrackResponseEntry 18 }

respSubject OBJECT-TYPE
 SYNTAX      DisplayString
```

```
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The full text of the subject of the tracked message"
 ::= { msgTrackResponseEntry 19 }
```

```
respMsgType OBJECT-TYPE
 SYNTAX      MsgType
 ACCESS     read-only
 STATUS      current
 DESCRIPTION
   "The type of the tracked message"
 ::= { msgTrackResponseEntry 20 }

-- CONFORMANCE INFORMATION
-- Used ONLY in V2 MIBs
messageTrackingConformance OBJECT IDENTIFIER ::= {
   mta-message-track 5 }
messageTrackingGroups          OBJECT IDENTIFIER ::= {
   messageTrackingConformance 1 }
messageTrackingCompliances OBJECT IDENTIFIER ::= {
   messageTrackingConformance 2 }


-- COMPLIANCE STATEMENTS
-- Used ONLY in V2 MIBs
limitedCompliance       MODULE-COMPLIANCE
 STATUS current
 DESCRIPTION
    "The basic levels of compliance for SNMPv2 entities that implement
this MIB
      for message tracking requiring the knowledge of a message Id."
 MODULE      -- this module
 MANDATORY-GROUPS { msgIdGroup }
 ::= { messageTrackingCompliances 1 }

basicCompliance       MODULE-COMPLIANCE
 STATUS current
 DESCRIPTION
    "The basic levels of compliance for SNMPv2 entities that implement
this MIB
      for message tracking without requiring the knowledge of a message
Id."
 MODULE      -- this module
 MANDATORY-GROUPS { msgIdGroup, basicGroup }
 ::= { messageTrackingCompliances 2 }

enhancedCompliance       MODULE-COMPLIANCE
 STATUS current
 DESCRIPTION
    "The basic levels of compliance for SNMPv2 entities that implement
this MIB
      for message tracking without requiring the knowledge of a message
```

```
Id and
      allowing an enhanced level of query capabilities."
 MODULE      -- this module
```

```
MANDATORY-GROUPS { msgIdGroup, basicGroup, enhancedGroup }
 ::= { messageTrackingCompliances 3 }

gatewayCompliance       MODULE-COMPLIANCE
 STATUS current
 DESCRIPTION
    "The basic levels of compliance for SNMPv2 entities that implement
this MIB
      for message tracking across mta's that perform a gateway
function."
 MODULE      -- this module
 MANDATORY-GROUPS { msgIdGroup, basicGroup, enhancedGroup,
                                                gatewayGroup }
 ::= { messageTrackingCompliances 4 }

-- UNITS OF CONFORMANCE
-- Used ONLY in V2 MIBs
msgIdGroup OBJECT-GROUP
 OBJECTS {
      msgTrackNextRequestIndex,  reqRowStatus,
      reqResponseStatus, reqMaxResponses,  reqUniqueMsgId,
      reqFailureReason,  respDispositionStatus,  respDispositionTime,
      respNonDeliveryReason, respMsgArrivalTime, respUniqueMsgId,
      respInboundOriginator, respInboundRecipient
 }
 STATUS current
 DESCRIPTION
    "  A collection of objects for tracking messages where the
messageId is
       known with responses containing basic message information."
    ::= { messageTrackingGroups 1 }

basicGroup OBJECT-GROUP
 OBJECTS {
      reqInboundOriginator, reqInboundRecipient, reqOriginatorNameForm,
      reqRecipientNameForm, reqEarliestArrivalTime,
reqLatestArrivalTime
 }
 STATUS current
 DESCRIPTION
    "  A collection of objects for tracking messages where the
messageId is not
known
                      with responses containing basic message
information."
    ::= { messageTrackingGroups 2}
```

```
enhancedGroup OBJECT-GROUP
 OBJECTS {
      reqSubject,           reqMinMsgSize,    reqMaxMsgSize,
      reqDispositionStatus,  reqMsgType,         reqCollapseRecipients,
      respMsgSize,              respMsgPriority,
respSupplementalInformation,
      respSubject,                 respMsgType
 }
 STATUS current
 DESCRIPTION
    " A collection of objects for tracking messages where the messageId
is not
      known with responses containing enhanced message information as
      well as enhanced query capabilities."
    ::= { messageTrackingGroups 3}

gatewayGroup OBJECT-GROUP
 OBJECTS {
      reqInboundMsgId,           reqOutboundMsgId,
reqOutboundOriginator,
      reqOutboundRecipient,   respNextHopMta,        respPrevHopMta,
      respInboundMsgId,         respOutboundMsgId,
respOutboundOriginator,
      respOutboundRecipient
 }
 STATUS current
 DESCRIPTION
    "  A collection of object for tracking messages that have passed
through a
       gateway mta."
    ::= { messageTrackingGroups 4}


END
```

4 Usages

A general model of message flow between MTAs has to be presented before a MIB can be described. Generally speaking, message flow occurs in three steps. We use the term "MTA" loosely to refer to any processing entity that is responsible for message delivery:

(1) Messages are received by the MTA from user agents, message stores, other

MTAs, and gateways.
(2) The "next hop" for the each message is determined. This is simply the
destination the message is to be transmitted to; it may or may not be the final
destination of the message. Next hop is established on a per-recipient basis.
Next hop may be a User Agent, a Message Store, another MTA, or a gateway.

(3) Information about the message and the next hop are written out to a log in some format. Messages are transmitted to the appropriate destination, which may be a user agent, message store, another MTA, or another gateway.

In terms of message tracking, the key step is step number 3, the logging of information. Once information about a message is written to a log, it can be subsequently queried by any number of means. It is the intent of this specification to support the following general message tracking usages.

Tracking an Individual Message

It is often necessary for messaging operations to track down the whereabouts and/or history of an individual message managed object. This is typically in response to a particular user tracking request regarding a previously sent message. It might also be used by messaging operations to verify that one or more message paths are operating properly.

Entire Path or Current Status

For some inquiries, it is desired that the entire path taken by the message to date be discovered, including the current disposition of the message (for one or more recipients). For others, it is necessary only to discover the current disposition of the message. Note however that it may be necessary to uncover its entire path to date in order to determine its current disposition. To discover a message's entire path, it will be necessary to query the Management Agent (MA) of each MTA and/or gateway encountered by the message, beginning with the originating MTA, to discover the status of the message from that MTA or gateway's perspective. Dispositions include: Transferred-to another MTA or gateway, In-queue within an MTA or gateway, Delivered, Non-delivered, Redirected, or Dlist-expanded. If a message was relayed to another MTA or gateway, then the MA of that MTA or gateway must be queried in turn. This must be repeated

until
the MTA or gateway which currently has responsibility for the message
is
reached.

To discover a message's current disposition, it might in fact be
necessary to track
the message's entire path, since typically the current status can only
be
discovered by a forward traversal through the message path.
Alternatively, to
minimize expense, the MA of any MTA/gateway which is estimated to be in
the
message path can be queried; if the message is shown to have reached
this
estimated point, then previous points in the path are irrelevant. Best
estimate
may therefore be the MA of the destination MTA, if known. Other
estimates
are made based on arbitrary intelligence which may be available.

For a message with multiple recipients, a different path/ disposition may be relevant for each recipient. Thus, message tracking queries must be made to discover the path/disposition of each recipient of the message, or for each recipient explicitly identified in the tracking request. For cases in which multiple recipient message copies are transferred-to the same MTA or gateway, these queries may be combined into a single query, for convenience.

Identifying Messages to Track

In some cases, the user may have retained some sort of a unique identifier for the message of interest. This will be true, for example, because an explicit trackability service was requested for the message, or because the user logs all outgoing messages. In such cases, a message query can be made simply on the basis of providing this unique identifier to the message tracking system.

Optionally, this query can be bounded by:

o specific recipient(s) of interest;
o time range of interest;
o or other bounding information.

In other cases, the user may only know some general descriptive information about the message. For example, the user may know who originated it, to whom it was sent, and roughly what time of day it was submitted. In such cases, it will be necessary to provide descriptive information to the message tracking system and allow it to respond with a list of matching messages which it knows about, along with their unique identifiers.

In either case, the query can specify or imply a list of information which it wishes the query response to return to the requester.

Query Responses

The query response must then return the requested information, or return an
indication as to why it cannot return the requested information. If the returned
information includes an indication that the message has been further relayed,
then a subsequent query must made to the transferred-to MTA or gateway.
Specific response information must be provided/implied in order to allow for
the creation of subsequent queries. For example:

o the name of the transferred-to MTA or gateway
o the name of the MA of the transferred-to MTA or gateway
o the names/addresses of the recipient(s) of interest which have taken this relay
path. Note that this may be a "redirected" recipient.
o If this response comes from a gateway, the translated versions of key identification fields.

This subsequent query can optionally be formulated automatically. See next section on "Automating Follow-up Queries" for details on follow-up query creation.

Automating Follow-up Queries

In order to support a fully automated path search, it is convenient and powerful for a management agent to be able to automatically make all of the queries required to discover the full path of an individual message. The MC can accomplish this by creating an initial query, and then creating a follow-on query utilizing information returned by the initial query, and repeating this process until the current disposition of the message is discovered.

Follow-on queries will need to be generated on a per-recipient basis. For cases in which multiple recipient message copies were transferred-to the same MTA or gateway, these queries may be combined into a single query, for efficiency.

In general, a follow-on query will need to:

o Be directed at the MA which corresponds to the MTA or gateway to which the message (copy) was relayed.
o Include the names/addresses of the recipient(s) which have taken this relay path. Note that this may be a "redirected" recipient.
o Include the same bounding criteria as did the initial query, with the addition of the unique message identification if that information was not previously available.
o If the just-completed query was to a gateway, then the new translated formats for the following will need to be used, if applicable:
  - Message ID
  - Recipient address
  - Originator address
  - Priority value
  - Size.

Tracking Aggregates of Messages

It is sometimes desirable for messaging operations to be able to

determine
information about the history/status of all messages which match a given set
of characteristics that have passed through their systems.

Identifying Messages to Track

In some cases, messaging operations may have retained some sort of a unique
identifier for the messages of interest. More probably, messaging operations is
simply interested in the set of messages which:

o went through its systems within a given time range, or
o were sent through its systems by a given originator, or
o were sent through its systems to a given recipient, or
o were sent through its systems with a given size or priority

In such cases, it will be necessary to provide descriptive information to the
message tracking system and allow it to respond with information on all of the
matching messages which it knows about. The query can specify or imply a list
of information which it wishes the query response to return to the requester.

Query Responses

The query response must then return the requested information, or return an
indication as to why it cannot return the requested information.

4.1 Security

[TBD]

5. Future Considerations

A version of this information is also available in the Management Information
Format (MIF) defined by the Desktop Management Task Force (DMTF).

As SNMP continues to evolve (e.g. SNMPv3),
the mechanisms in this document will leverage those new technologies.

6. Acknowledgements

This draft also incorporates the intellectual contributions of

Bruce Greenblatt
Niraj Jain
Sue Lebeck
Roger Mizumori
Edward Owens

7. References

[1] Case, J., McCloghrie, K., Rose, M., and S. Waldbusser, "Structure of Management Information for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1902, SNMP Research,Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, September 1996.

[2] McCloghrie, K., and M. Rose, Editors, "Management Information Base for Network Management of TCP/IP-based internets: MIB-II", STD 17, RFC 1213, Hughes LAN Systems, Performance Systems International, March 1991.

[3] Case, J., McCloghrie, K., Rose, M., and S, Waldbusser, "Protocol Operations for version 2 of the Simple Network Management Protocol (SNMPv2)", RFC 1905, SNMP Research,Inc., Hughes LAN Systems, Dover Beach Consulting, Inc., Carnegie Mellon University, September 1996.

[MODEL] Case, J., McCloghrie, K., Rose, M., and S, Waldbusser, "Protocol Operations for version 2 of the Simple Network Management

Authors' Addresses

Bruce Ernst
Lotus Development Corporation

Phone: +1 610 251 3404
E-Mail: bruce_ernst@lotus.com

Gordon Jones
MITRE Corporation
1820 Dolley Madison Blvd.
McLean, VA 22102-3481

Phone: +1 703 883 7670
E-Mail: gbjones@mitre.org

Jonathan Weintraub
Electronic Data Systems, Inc.

+1 301 619 3101