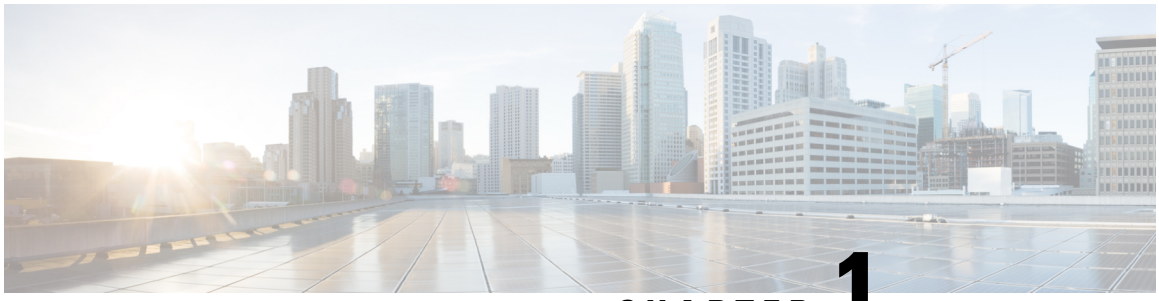
Cisco Unity Connection Messaging Interface (CUMI) API

Americas Headquarters
Cisco Systems, Inc.
170 West Tasman Drive
San Jose, CA 95134-1706
USA
http://www.cisco.com
Tel: 408 526-4000
800 553-NETS (6387)
Fax: 408 527-0883

CONTENTS



CHAPTER 1

Cisco Unity Connection Messaging Interface (CUMI) API

Links to Other API pages: Cisco Unity Connection APIs

Introduction

The Cisco Unity Connection Messaging Interface (CUMI) is a messaging API for Cisco Unity Connection that has been designed to be stable and simple to use. It is based on leading industry standards for web-based API development, and provides access to a wide set of Connection messaging functionality.

If you are a customer or developer who needs the ability to integrate Connection into an existing enterprise-wide portal, CUMI offers a secure method for doing the following:

- Sending messages
- Receiving messages
- Replying to messages
- Sending broadcast messages (provided the user account is enabled to send broadcast messages)
- Sending dispatch messages
- Receiving notifications of new messages

Note By default, API access to playback secure messages is turned off. To enable API access to secure messages, check the Allow Access to Secure Message Recordings Through the Cisco Unity Connection Messaging Interface (CUMI) setting on the System Settings > Advanced > API Settings page in Cisco Unity Connection Administration.

Note All the above functions associated with CUMI API support both the IPv4 and IPv6 addresses. However, the IPv6 address works only when Connection platform is configured in Dual (IPv4/IPv6) mode.

Note With Cisco Unity Connection 9.1(1), the single sign-on feature is enabled for all the Connection Rest APIs. For more information, see the "Single Sign-On in Cisco Unity Connection" chapter in Security Guide for Cisco Unity Connection 9.x https://www.cisco.com/c/en/us/td/docs/voice_ip_comm/connection/9x/security/guide/9xcucsecx/9xcucsec061.html

Beginning with Cisco Unity Connection 10.5 and later, when one or more tenants are configured on a single installation of Cisco Unity Connection, a user with **Mailbox Access Delegate Account** role and belonging to a particular tenant will be able to access messages of all the users within the same tenant only.

How the documentation is organized

- **Accessing Mailboxes and Folders**
- **Sending Messages using the API**
- **Message Recall API**
- **Working with Notifications**
- **Working with Dispatch Messages**
- **Working with Broadcast Messages**
- **Special Features**
- **Samples**

Technical Details

CUMI is a REST interface that standardizes operations such as add, delete, and modify. The XML comes with standard XML schema definitions that are annotated with information about what is in them.

As a web-based interface, CUMI is independent of operating system and programming language and does not require any client libraries to use.

Getting started

In order to begin developing with the Cisco Unity Connection CUMI API, you will need to obtain the following:

Hardware

- Cisco Media Convergence Server (MCS) for Cisco Unity Connection version 12.x and later

- For detailed hardware requirements, see the Cisco Unity Connection 8.x Supported Platforms List

Software

- Cisco Unity Connection Software Ordering
- Not for Resale Kits (Must be eligible to purchase)
- Select Unified Communications System Release Kit

Discounts for some of the required hardware and software may be available for participants in the Cisco Technology Developer Program.

We recommend that all developers have an up-to-date Cisco Developer Services support agreement. This provides the developer with access to professional support and assistance for application development.

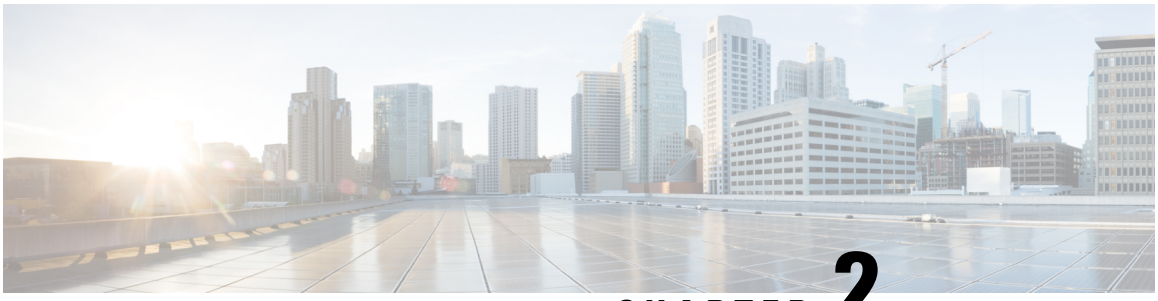
Other CUMI Resources

Additional information about CUMI is also available on the Cisco Developer Network. Note, however, that the documentation here on the DocWiki is the most up-to-date documentation available for CUMI.

Troubleshooting

See the following for information on troubleshooting all Connection APIs:

Troubleshooting (applies to all Connection APIs)



CHAPTER 2

Cisco Unity Connection Messaging Interface (CUMI) API -- Using the CUMI API

About Mailboxes and Folders

The root of Cisco Unity Connection Messaging Interface (CUMI) API is the Mailbox resource that is associated with each user. This contains some general information about the user's mailbox, and also contains a Folders resource that lists the folders for the mailbox. This list is currently fixed, although it is possible that folders may be added in the future.

Beginning with Cisco Unity Connection 10.5 and later, when one or more tenants are configured on a single installation of Cisco Unity Connection, a user with **Mailbox Access Delegate Account** role and belonging to a particular tenant will be able to list messages of all the users within the same tenant only.

GET operation on the Mailbox returns properties of the mailbox (for example, quotas) as well as a link to the Folders element for the mailbox:

```
GET /vmrest/mailbox
<xs:complexType name="Mailbox">
<xs:all>
<xs:element name="DisplayName" type="xs:string" />
<xs:element name="CurrentSizeInBytes" type="xs:long" />
<xs:element name="IsPrimary" type="xs:boolean" />
<xs:element name="IsStoreMounted" type="xs:boolean" />
<xs:element name="IsStoreOverFlowed" type="xs:boolean" />
<xs:element name="IsMailboxMounted" type="xs:boolean" />
<xs:element name="IsWarningQuotaExceeded" type="xs:boolean" />
<xs:element name="IsReceiveQuotaExceeded" type="xs:boolean" />
<xs:element name="IsSendQuotaExceeded" type="xs:boolean" />
<xs:element name="WarningQuota" type="xs:long" />
<xs:element name="ReceiveQuota" type="xs:long" />
<xs:element name="SendQuota" type="xs:long" />
<xs:element name="IsDeletedFolderEnabled" type="xs:boolean" />
<xs:element name="FoldersURI" type="xs:anyURI" />
</xs:all>
</xs:complexType>
```

Mailbox Folder Operations

There are three folders currently supported on a Unity Connection Mailbox -

- Inbox
- Sent items
- Deleted items

Performing GET operation on the folders returns the fixed list of folders:

```
GET /vmrest/mailbox/folders
```

Following are the properties associated with each folder:

- DisplayName
- MessageCount
- Unique Serial Number (Unity Connection 11.5 and later)
- UIDValidity (Unity Connection 11.5 and later)

A GET operation on a folder returns the associated properties.

```
GET /vmrest/mailbox/folders/<folder_name>
```

Following is the response of above GET request.

```
<Folder>
<DisplayName></DisplayName>
<MessageCount></MessageCount>
<USN></USN>
<UIDValidity></UIDValidity>
</Folder>
```

Note Each time an operation is performed on Inbox folder or Delete folder, the Unique Serial Number (USN) of both the folders changes.

Inbox Folder Operations

All folder operations can be executed by a user when connecting with his/her credentials. When using the administrative credentials, using the userobjectid parameter will allow administrators to do the same operations on the users mailbox.

A GET operation on a folder returns a message list:

```
GET /vmrest/mailbox/folders/inbox/messages
```

Response:

```
<Messages total="1">
  <Message>
    <Subject>Message from user1 (1001)</Subject>
    <Read>true</Read>
    <Dispatch>false</Dispatch>
    <Secure>false</Secure>
    <Priority>Normal</Priority>
    <Sensitivity>Normal</Sensitivity>
    <SessionId>122162627c12b61</SessionId>
    <URI>/vmrest/messages/0:b4ffbaa5-19ad-4f4f-b95a-933a32e00d64</URI>
    <MsgId>0:b4ffbaa5-19ad-4f4f-b95a-933a32e00d64</MsgId>
    <From>
      <DisplayName>user1</DisplayName>
      <SmtpAddress>connectionserver</SmtpAddress>
      <DtmfAccessId>1001</DtmfAccessId>
    </From>
    <CallerId>
      <CallerNumber>1001</CallerNumber>
      <CallerName>user1</CallerName>
    </CallerId>
    <ArrivalTime>1522069679000</ArrivalTime>
    <Size>37291</Size>
    <Duration>3340</Duration>
    <IMAPUid>22</IMAPUid>
    <FromSub>true</FromSub>
    <MsgType>Voice</MsgType>
  </Message>
</Messages>
```

A GET operation on messages in inbox folder returns a particular message by providing messageid :

```
GET/vmrest/messages/<messageid>
```

Response:

```
<Message>
  <Subject>Message from user1 (1001)</Subject>
  <Read>true</Read>
  <Dispatch>false</Dispatch>
  <Secure>false</Secure>
  <Priority>Normal</Priority>
  <Sensitivity>Normal</Sensitivity>
  <Attachments>
    <Attachment>
      <URI>/vmrest/messages/0:b4ffbaa5-19ad-4f4f-b95a-933a32e00d64/attachments/0</URI>
      <contentType>audio/wav; name=voicemessage.wav</contentType>
      <contentTransferEncoding/>
      <contentDisposition>inline; filename=VoiceMessage.wav;
voice=Voice-Message</contentDisposition>
    </Attachment>
  </Attachments>
  <SessionId>122162627c12b61</SessionId>
  <Recipients>
    <Recipient>
      <Type>TO</Type>
      <Address>
        <DisplayName/>
        <SmtpAddress>connectionserver</SmtpAddress>
        <DtmfAccessId>1009</DtmfAccessId>
      </Address>
    </Recipient>
  </Recipients>
  <URI>/vmrest/messages/0:b4ffbaa5-19ad-4f4f-b95a-933a32e00d64</URI>
  <MsgId>0:b4ffbaa5-19ad-4f4f-b95a-933a32e00d64</MsgId>
  <From>
    <DisplayName>user1</DisplayName>
    <SmtpAddress>user1@ucbu-aricent-vm550.cisco.com</SmtpAddress>
    <DtmfAccessId>1001</DtmfAccessId>
  </From>
  <CallerId>
    <CallerNumber>1001</CallerNumber>
    <CallerName>user1</CallerName>
  </CallerId>
  <ArrivalTime>1522069679000</ArrivalTime>
  <Size>37291</Size>
  <Duration>3340</Duration>
  <IMAPUid>22</IMAPUid>
  <FromSub>true</FromSub>
  <MsgType>Voice</MsgType>
</Message>
```

A user with administrative privileges can list the messages of another users folder by passing the userobjectid of the other user:

```
GET /vmrest/mailbox/folders/inbox/messages?userobjectid=<userobjectid>
```

A user with administrative privileges can list the messages of another users folder having USN value greater than specified and by passing the userobjectid of the other user:

```
GET /vmrest/mailbox/folders/inbox/messages?userobjectid=<userobjectid>&usngreaterthan=<value>
```

A PUT operation on messages in Inbox folder can update the Subject or the Read field of the Messages. No other parameter of a message can be changed:

```
PUT /vmrest/messages/<messageid>
<Message>
     <Subject>New subject</Subject>
</Message>
```

A DELETE operation on messages in Inbox folder can delete the message from the folder. Whether the message is soft or hard deleted is dependent on the settings of the system.

```
DELETE /vmrest/messages/<message-id>
```

A hard or soft delete can be forced by passing in the harddelete parameter:

```
DELETE /vmrest/messages/<message-id>?userobjectid=<userobjectid>&harddelete=<true or false>
```

Sent Items Folder Operations

All folder operations can be executed by a user when connecting with his/her credentials. When using the administrative credentials, using the userobjectid parameter will allow administrators to do the same operations on the users mailbox.

A GET operation on the folder returns a message list:

```
GET /vmrest/mailbox/folders/sent/messages
```

A user with administrative privileges can list the messages of another users folder by passing the userobjectid of the other user:

```
GET /vmrest/mailbox/folders/sent/messages?userobjectid=<userobjectid>
```

A user with administrative privileges can list the messages of another users folder having USN value greater than specified and by passing the userobjectid of the other user:

```
GET /vmrest/mailbox/folders/sent/messages?userobjectid=<userobjectid>&usngreaterthan=<value>
```

A PUT operation on messages in the Sent Items folder can update the Subject of messages. No other parameter on a message can be changed:

```
PUT /vmrest/messages/<messageid>
<Message>
     <Subject>New subject</Subject>
</Message>
```

A DELETE operation on messages in the Sent Items folder can delete the message from the folder. Whether the message is soft or hard deleted is dependent on the settings of the system.

```
DELETE /vmrest/messages/<message-id>
```

A hard or soft delete can be forced by passing in the harddelete prarameter:

```
DELETE /vmrest/messages/<message-id>?userobjectid=<userobjectid>&harddelete=<true or false>
```

Deleted Items Folder Operations

All folder operations can be executed by a user when connecting with his/her credentials. When using the administrative credentials, using the userobjectid parameter will allow administrators to do the same operations on the users mailbox.

A GET operation on the folder returns a message list:

```
GET /vmrest/mailbox/folders/deleted/messages
```

A user with administrative privileges can list the messages of another users folder by passing the userobjectid of the other user:

```
GET /vmrest/mailbox/folders/deleted/messages?userobjectid=<userobjectid>
```

A user with administrative privileges can list the messages of another users folder having USN value greater than specified and by passing the userobjectid of the other user:

```
GET
/vmrest/mailbox/folders/deleted/messages?userobjectid=<userobjectid>&usngreaterthan=<value>
```

A PUT operation on messages in the Deleted Items folder to update the Subject on the messages. No other parameter on a message can be changed:

```
PUT /vmrest/messages/<messageid>
<Message>
     <Subject>New subject</Subject>
</Message>
```

A DELETE operation on messages in the Deleted Items folder will delete the message from the folder. This is a hard (permanent) delete.

```
DELETE /vmrest/messages/<message-id>
```

A POST operation on the Deleted Items folder can be used to empty the whole folder. The messages are hard deleted.

```
POST /vmrest/mailbox/folders/deleted/messages?method=empty
```

Offset and Limit

Each of the folders will accept the parameters "pagenumber" and "rowsperpage" to specify which messages to retrieve:

```
/vmrest/mailbox/folders/inbox/messages?pagenumber=1&rowsperpage=10
```

Sorting

Initially, server-side sorting will be limited to what can be done efficiently by the database, and will default to placing new messages first, followed by read messages, and sorted within each by ArrivalTime.

As recommended in the VTG REST guidelines, sorting will be controlled via "sortkey" and "sortorder" parameters, although initially only the following sort orders will be supported by the server:

Sort Description	Sort Parameters

Newest first	`no parameters (default) or sortkey=arrivaltime&sortorder=descending`
Oldest first	`sortkey=arrivaltime&sortorder=ascending`
Urgent first	`sortkey=priority&sortorder=descending`

Filtering

Filtering can be done on the folders by read, priority, dispatch, type, and USN of the message.

```
read={true|false}
priority={urgent|normal|low}
dispatch={true|false}
type={voice|fax|email|receipt}
usngreaterthan={Integer}
```

Examples

To get a list of unheard voice messages:

```
GET /vmrest/mailbox/folders/inbox/messages?read=false&type=voice
```

To get a list of unheard urgent messages:

```
GET
/vmrest/mailbox/folders/inbox/messages?read=false&priority=urgent
```

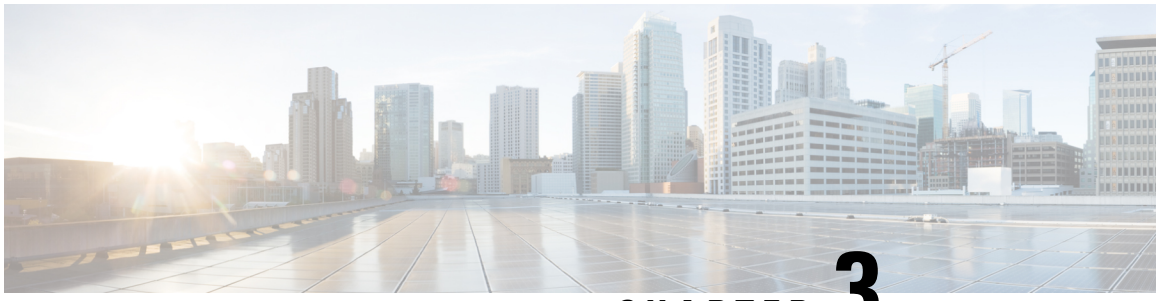
To get a list of saved (deleted) messages:

```
GET /vmrest/mailbox/folders/deleted/messages
```

To get a list of messages with USN greater than 10:

```
GET /vmrest/mailbox/folders/inbox/messages?usngreaterthan=10
```

Examples



CHAPTER 3

Cisco Unity Connection Messaging Interface (CUMI) API -- Using the CUMI API for Sending Messages

About Messages

A Message resource is what you might think of as a message "header" or "envelope." It is a small description of the message (from, to, time, etc.), so that loading a list of messages is efficient; the actual message content is provided in the form of links to one or more attachments.

There is no fixed limit on the number of recipients a message can have. Therefore, the recipient list is also provided as a link to an attachment; because there is no limit on the number of recipients, loading recipients into the Message resource could have a big impact on performance.

Non-delivery receipts (NDRs) have one additional child resource, FailedRecipients, which includes a Recipient URI and a failure status code. New attribute "FutureDelivery" can be set while sending voicemails at specified time in future.

Beginning with Cisco Unity Connection 10.5 and later, when one or more tenants are configured on a single installation of Cisco Unity Connection, a user with **Mailbox Access Delegate Account** role and belonging to a particular tenant will be able to send messages of all the users within the same tenant only.

Sending Messages

A message can be sent by a POST request to the root URI for messages. The content of the request is "mutipart/form-data". There must be three pieces of data in the request in the following order:

1. a message in either "application/xml" or "application/json" format
2. a list of recipients
3. auto data as "audio/wav" (optional as a message can be sent with no audio)

Schema

```
<xs:simpleType name="priorityType">
<xs:restriction base="xs:string">
<xs:enumeration value="Low" />
<xs:enumeration value="Normal" />
<xs:enumeration value="Urgent" />
</xs:restriction>
</xs:simpleType>
<xs:simpleType name="sensitivityType">
<xs:restriction base="xs:string">
<xs:enumeration value="Normal" />
<xs:enumeration value="Personal" />
<xs:enumeration value="Private" />
<xs:enumeration value="Confidential" />
</xs:restriction>
</xs:simpleType>
<xs:complexType name="callerIdType">
<xs:all>
<xs:element name="CallerNumber" type="xs:string" />
<xs:element name="CallerName" type="xs:string" />
<xs:element name="CallerImage" type="xs:anyURI" />
</xs:all>
</xs:complexType>
<xs:simpleType name="messageType">
<xs:restriction base="xs:string">
<xs:enumeration value="Email" />
<xs:enumeration value="DR" />
<xs:enumeration value="Voice" />
<xs:enumeration value="RR" />
<xs:enumeration value="Fax" />
<xs:enumeration value="NDR" />
</xs:restriction>
</xs:simpleType>
<xs:complexType name="Message">
<xs:all>
<!-- fields that can be modified by the client at any time -->
<xs:element name="Subject" type="xs:string" minOccurs="0" />
<xs:element name="Read" type="xs:boolean" minOccurs="0" />
<!-- flags that can be set by the client on send -->
<xs:element name="Dispatch" type="xs:boolean" minOccurs="0" />
<xs:element name="Secure" type="xs:boolean" minOccurs="0" />
<xs:element name="Priority" type="priorityType" minOccurs="0" />
<xs:element name="Sensitivity" type="sensitivityType" minOccurs="0" />
<xs:element name="ReadReceiptRequested" type="xs:boolean" minOccurs="0" />
 <!-- New attribute is addded for Future Delivery  -->
<xs:element name="FutureDelivery" type="xs:string"/>
<xs:element name="DeliveryReceiptRequested" type="xs:boolean" minOccurs="0" />
<!-- fields that are only filled out when getting individual message details -->
<xs:element name="Attachments" type="Attachments" minOccurs="0" />
<xs:element name="Recipients" type="Recipients" minOccurs="0" />
<!-- message envelope elements -->
<xs:element name="URI" type="xs:anyURI" />
<xs:element name="MsgId" type="xs:string" />
<xs:element name="From" type="Address" />
<xs:element name="CallerId" type="callerIdType" />
<xs:element name="ArrivalTime" type="xs:long" />
<xs:element name="Size" type="xs:unsignedLong" />
<xs:element name="Duration" type="xs:unsignedLong" />
<xs:element name="IMAPUid" type="xs:string" />
<xs:element name="FromSub" type="xs:boolean" />
<xs:element name="FromVmIntSub" type="xs:boolean" />
<xs:element name="Flagged" type="xs:boolean" minOccurs="0" />
<xs:element name="MsgType" type="messageType" minOccurs="0" />
<xs:element name="ModificationTime" type="xs:long" minOccurs="0" />
<xs:element name="IsDraft" type="xs:boolean" minOccurs="0" />
```

```
<xs:element name="IsDeleted" type="xs:long" minOccurs="0" />
<xs:element name="IsSent" type="xs:long" minOccurs="0" />
<xs:element name="IsFuture" type="xs:long" minOccurs="0" />
<xs:element name="FolderURI" type="xs:anyURI" minOccurs="0" />
</xs:all>
</xs:complexType>
<xs:element name="Message" type="Message" />
<xs:element name="Messages">
<xs:complexType>
<xs:sequence>
<xs:element name="Message" type="Message" maxOccurs="unbounded" />
</xs:sequence>
</xs:complexType>
</xs:element>
```

When sending a message, the following fields can be set:

- dispatch
- secure
- priority
- sensitivity
- read receipt requested
- delivery receipt requested
- future delivery attribute

On a received message, the following fields can be modified:

- priority
- subject
- read

Although a message can appear in a folder, it is always available via a unique id:

```
/vmrest/mailbox/messageid
```

Because messages include attachments and recipients that are not efficiently included in the envelope of the message, they are filled out only when an individual message is requested.

The following is the resource definition for attachments and recipients:

```
<xs:complexType name="Attachment">
 <xs:all>
 <xs:element name="URI" type="xs:anyURI" />
 </xs:all>
 </xs:complexType>
 <xs:element name="Attachment" type="Attachment" />
 <xs:complexType name="Attachments">
 <xs:sequence>
 <xs:element name="Attachment" type="Attachment" maxOccurs="unbounded" />
 </xs:sequence>
 </xs:complexType>
```

Recipients for a message are identified in a recipient list. Each recipient has a recipient type and an address. The address information for an inbound message is filled out with any information that is available. Each address that is part of a recipient for an outbound message must contain an object identifier and a type, or a "blind" address in the SmtpAddress field. User needs to specify either an object identifier or DtmfAccessID or SmtpAddress in order to sendMessage to another user.

The schema for recipients and addresses:

```
<xs:simpleType name="addressType">
 <xs:restriction base="xs:string">
 <xs:enumeration value="SUBSCRIBER" />
 <xs:enumeration value="DISTRIBUTIONLIST" />
 <xs:enumeration value="PRIVATELIST" />
 <xs:enumeration value="CONTACT" />
 <xs:enumeration value="VPIMCONTACT" />
 </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="Address">
 <xs:all>
 <xs:element name="ObjectId" type="xs:string" minOccurs="0" />
 <xs:element name="Type" type="addressType" minOccurs="0" />
 <xs:element name="UserGuid" type="xs:string" minOccurs="0" />
 <xs:element name="DisplayName" type="xs:string" minOccurs="0" />
 <xs:element name="SmtpAddress" type="xs:string" minOccurs="0" />
 <xs:element name="DtmfAccessID" type="xs:string" minOccurs="0" />
 </xs:all>
 </xs:complexType>
 <xs:element name="Address" type="Address" />
 <xs:complexType name="Addresses">
 <xs:sequence>
 <xs:element name="Address" type="Address" maxOccurs="unbounded" />
 </xs:sequence>
 </xs:complexType>
 <xs:element name="Addresses" type="Addresses" />
 <xs:simpleType name="recipientType">
 <xs:restriction base="xs:string">
 <xs:enumeration value="TO" />
 <xs:enumeration value="CC" />
 <xs:enumeration value="BCC" />
 </xs:restriction>
 </xs:simpleType>
 <xs:complexType name="Recipient">
 <xs:all>
 <xs:element name="Type" type="recipientType" />
 <xs:element name="Address" type="Address" />
 </xs:all>
 </xs:complexType>
 <xs:element name="Recipient" type="Recipient" />
 <xs:complexType name="Recipients">
 <xs:sequence>
 <xs:element name="Recipient" type="Recipient" maxOccurs="unbounded" />
 </xs:sequence>
 </xs:complexType>
```

Container objects are simply attachments and recipients.

Addressing

Addressing in CUMI consists of a lookup method that returns a list of addresses matching a specified name or the beginning of a name:

URI	Request	Description
/mailbox/addresses?name=<full or partial name>	GET	Returns an Address resource that lists all of the matched names (up to a maximum of 100).

The name can be a whole name, or the beginning of the first name, last name, display name, or alias. The name value in the URI will match distribution lists, personal distribution lists, contacts, and users.

For example, when searching for the user "Alexander Bell," the name could be any of the following:

```
name=abell
 name=alexander
 name=alex
 name=bell
 name=bel
 name=alexander be
 name=alex be
```

Error Handling

When errors are encountered on the server, the CUMI API adheres to the standard REST practice of using the closest HTTP code available. It also follows the recommendation in "RESTful Web Services," and returns an error document that is human readable (and also program readable).

The approach is to return codes in descending order: first the HTTP code, then the relevant VMWS error if possible, and finally the original error code and message. We generally have two levels of errors in this particular case, but the errors will be a general list to provide a more extensible mechanism.

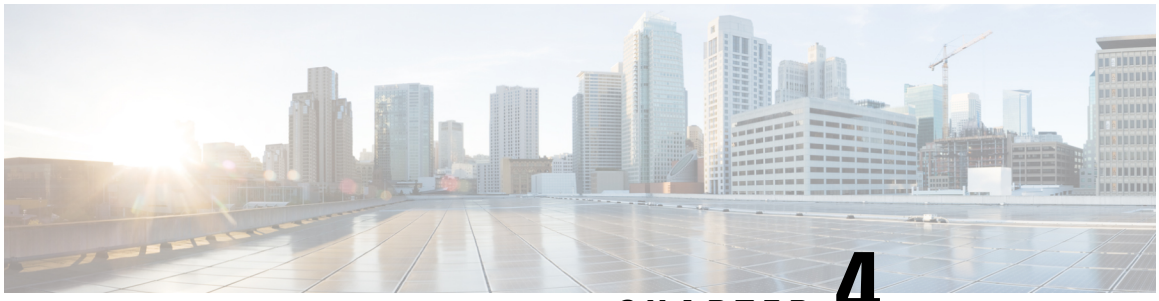
For example:

```
403 Forbidden
 Content-Type: application/xml
 Date: Fri, 10 Nov 2008 20:04:45 GMT
 Server: UnityConnection
 Transfer-Encoding: chunked
 <?xml version="1.0" encoding="UTF-8"?>
 <Error>
 <Code>E_VMWS_MSG_MAILBOX_FULL</Code>
 <Message>Destination mailbox is full or send quota was exceeded.</Message>
 <Code>0x80046408</Code>
 <Message>E_CML_SEND_QUOTA_EXCEEDED</Message></Error>
```

The VMWS error codes are intended to make it easier for Cannonball and Midlet clients to cut over to this interface. In addition, having a code between HTTP and the lowest-level component error code is useful. New VMWS errors that do not apply to the SOAP interface are added in the form "E_VM{_}REST__<new-error>".

Here's the XSD snippet for a RestError:

```
<xs:element name="RestError">
 <xs:complexType>
 <xs:all>
 <xs:element name="error" maxOccurs="unbounded">
 <xs:complexType>
 <xs:all>
 <xs:element name="code" type="ErrorCode"/>
 <xs:element name="message" type="xs:string"/>
 </xs:all>
 </xs:complexType>
 </xs:element>
 </xs:all>
 </xs:complexType></xs:element>
```



CHAPTER 4

Cisco Unity Connection Messaging Interface (CUMI) API -- Sending a Voice Message with One or More Attachments

About Sending Voice Messages with One or More Attachments

In Cisco Unity Connection 8.5 and later, CUMI will allow voice mails to be sent that have additional attachments of any type. (In earlier versions of Connection, CUMI allows only voice messages with one audio attachment to be sent.)

-

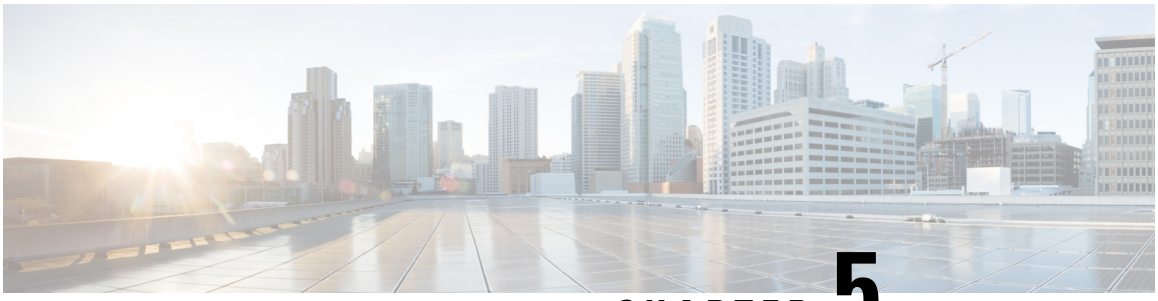
Sending a Voice Message with Attachments

The basics of sending a voice message involve a creating multipart HTTP form that contains an object representation of a Message, an object representation of a Recipient list and a set of zero or more additional parts that will be added as message attachments. If there are parts beyond the first two, then the third one must be either audio/wav data or an object representation of a CallControl object (the CallControl object is used when the stream resides on the server after using CUTI to record over the phone). The third part is considered to be the "primary" body of the voice message.

The parts beyond the third part are passed transparently to the message system to be added as an attachment. These additional parts must have a content type and some data.

HTTP example for a send message POST request:

```
POST /vmrest/messages HTTP/1.1
 Content-Type: multipart/mixed;boundary=Boundary_1_25804854_1282226936453
 Accept: application/xml
 MIME-Version: 1.0
 User-Agent: Java/1.6.0_21
 Host: cuc-install-67.cisco.com
 Connection: keep-alive
 Content-Length: 489122
 --Boundary_1_25804854_1282226936453
 Content-Type: application/json
 {"Subject":"subscriber send message test","ArrivalTime":"0","FromSub":"false"}
 --Boundary_1_25804854_1282226936453
 Content-Type: application/json
 {"Recipient":{"Type":"TO","Address":
 {"ObjectId":"cffc8051-1374-4e57-8d83-9e25aff4385a","Type":"SUBSCRIBER"}}}
 --Boundary_1_25804854_1282226936453
 Content-Type: audio/wav
 RIFFWAVEfmt @>data
 ""
 <snip>
 --Boundary_1_25804854_1282226936453
 Content-Type: text/plain;name=textfile.tx
 This is a test text message
 A third line
 end on 5
 --Boundary_1_25804854_1282226936453
 Content-Type: audio/wav
 RIFFÀ¹
 <snip>
 --Boundary_1_25804854_1282226936453
 Content-Type: text/plain;name=textfile.txt
 This is a test text message
 A third line
 end on 5
 --Boundary_1_25804854_1282226936453--
```



CHAPTER 5

Cisco Unity Connection Messaging Interface (CUMI) API -- Message Recall API

Message Recall API

About Recalling Messages

In Cisco Unity Connection 10.0 and later, CUMI allows recalling of voice mails until the sent messages is available in sent folder.

Recalling Message using API

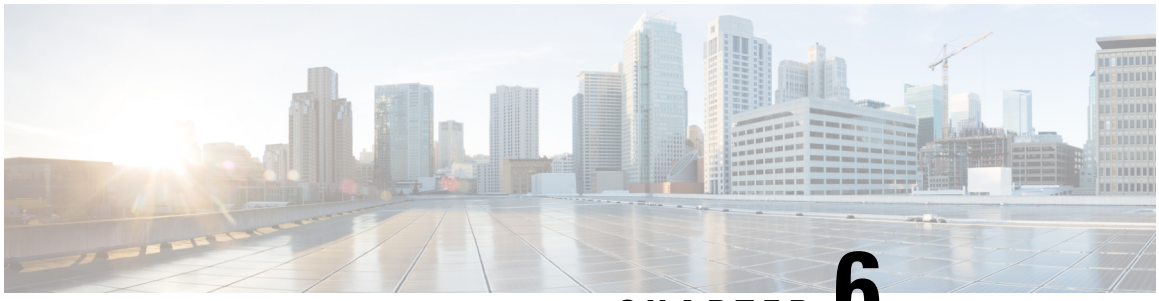
CUMI API allows recalling of voicemails using API. If messages is read by another user then this message will not get recalled from that user. If voice message is available in sent folder then voice message can be recalled and total count of voicemails for that user will be reduced. A message can be recall by a PUT request to the root URI:

```
/vmrest/messages/<message-id>/recall
```

```
Response:  Messages is successfully recalled
HTTP1.1  204
```

Example of API:

```
https://<ser/vmrest/messages/0:a8deea61-c2e3-4896-ad0a-124d1723ddcc/recall
```



CHAPTER 6

Cisco Unity Connection Messaging Interface (CUMI) API -- Using the CUMI API for Sending Notifications

About Notifications

CUMI supports Comet notifications for message operations on the Inbox and Deleted folders.

-

Getting Started

These are the basic steps to get up and running with Comet:

1. **Start Jetty:** Activate the Connection Jetty service by browsing to the Cisco Unity Connection Serviceability page and selecting Services. Find Connection Jetty in the list and select the Activate button.
2. **Request Notifications:** Clients need to request notifications so that the Connection server knows it should start publishing Comet events for the current authenticated user:

```
$.ajax({
 type: "POST",
 contentType: "application/xml; charset=utf-8",
 url: "/vmrest/mailbox?method=requestnotification",
 data: "{}",
 dataType: "text",
 success: function(subscriptionId) {
 gSubscriptionId = subscriptionId;
 alert("Requested events for mailbox, subscriptionId=" + subscriptionId);
 }
 });
```

This returns a subscriptionId that must be used to get Comet notifications for this user (rather than the userID, for security reasons).

3. **Subscribe for events:** This is just standard Comet now, but there are a few details to know such as the port and URL to use. Jetty runs on port 7080 for non SSL and port 7443 over SSL, and cometd is running at /vmevents/cometd. Here's sample code to initialize and subscribe (in JQuery):

```
$.comet.init("http://connection-server:7080/vmevents/cometd");
$.comet.subscribe("/vmrest/mailbox/" + gSubscriptionId, function(e)
{
alert("Event: " + e.data.EventType );
});
```

Comet Event Structure

Each Comet event has the following attributes:

DisplayName	Display Name of Mailbox
SubscriptionId	Subscription ID
USN	Message USN
EventTime	Time of event
EventType	Type of event
MailboxId	Mailbox ID
MessageInfo	Array of message info objects(see below)

MessageInfo is an array of one or more objects with the following attributes:

- MessageId
- CallerAni
- MsgType
- Priority
- ReceiveTime
- Sender

EventType will be one of the following:

EventType	Description
NEW_MESSAGE	This event is sent when a new message arrives in the inbox folder.
SAVED_MESSAGE	This message is sent when a message is marked as read in the inbox folder.
UNREAD_MESSAGE	This event is sent when a message is marked as unread in the inbox folder.

DELETED_MESSAGE	This event is sent when a message is deleted from the inbox folder.
DELETED_MESSAGE_CREATE	This event is sent when a new message arrives in the deleted folder.
DELETED_MESSAGE_READ	This event is sent when a message is marked as read in the deleted folder.
DELETED_MESSAGE_UNREAD	The event is sent when a message is marked as unread in the deleted folder.
DELETED_MESSAGE_DELETE	The event is sent when a message is hard deleted from the deleted folder or from the Inbox folder
DELETED_MESSAGE_UNDELETE	This event is sent when a message is marked as undeleted from the Deleted folder.

Example Comet Notifications

Example 1: When a message is deleted from inbox folder or when a new message arrives in the deleted folder, the following events take place:

- DELETED_MESSAGE
- DELETED_MESSAGE_CREATE

Note The USN remains the same for both the notification events.

Example 2: When a message is marked as undeleted from deleted folder, the following events take place:

- SAVED_MESSAGE or UNREAD_MESSAGE
- DELETED_MESSAGE_UNDELETE

Note The USN remains the same for both the notification events.

Example 3:

When a message is marked as hard deleted from the inbox folder, the following events take place:

- DELETED_MESSAGE
- DELETED_MESSAGE_CREATE
- DELETED_MESSAGE_DELETE

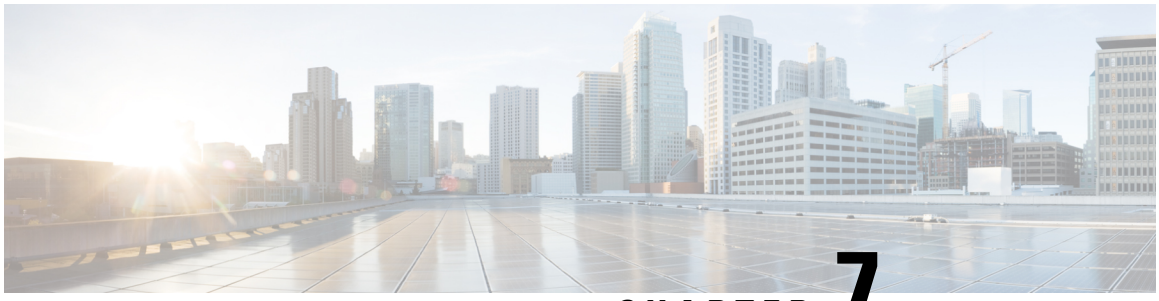
Note The USN remains the same for the first two notification events.

Apart from the above examples, the USN is not the same for any of the other notification scenarios.

Note: During the time Notifier service is down or inactive, if there is any action taken on a message that is either in the Inbox or Deleted folder, the notification for the latest status of that message is sent when the service is back to its active state.

Example Comet Event

```
{
 "MessageInfo": [ {
 "CallerAni": "null",
 "Sender": "null",
 "MessageId": "72204bd7-e5c3-446e-adb6-ae5f80db26fb",
 "ReceiveTime": "04:15:40 PM 10/30/2008",
 "Priority": "Normal-Priority",
 "MsgType": "Voice"
 }],
 "USN": 2265,
 "EventTime": "11:15:40 PM 10/30/2008",
 "SubscriptionId": "00bdcfd3-159a-48d3-ac7b-2f3b4f83db6c",
 "MailboxId": "abell",
 "EventType": "SAVED_MESSAGE",
 "DisplayName": "Alexander Bell"
 }
```



CHAPTER 7

Cisco Unity Connection Messaging Interface CUMI API -- Scaling Applications Using Notifications

About Notifications

Cisco Unity Connection Provides two types of Notifications

- COMET Notifications. Available through the CUMI. It is a subscription to an individual mailbox's events.
- Bulk Notification via the CUNI. The subscriptions can be created for multiple mailboxes at the same time.

Scalability

Cisco has tested the notification APIs under load to provide guidance on how they can be scaled. It is advised that development teams created solutions using these APIs validate their appliactions as the numbers can change based on the hardware performance as well as other loads on the system.

Test Hardware

The results are based on tests done on a 20,000 user OVA Virtual machine.

-

Test Results

CUMI COMET notifications only

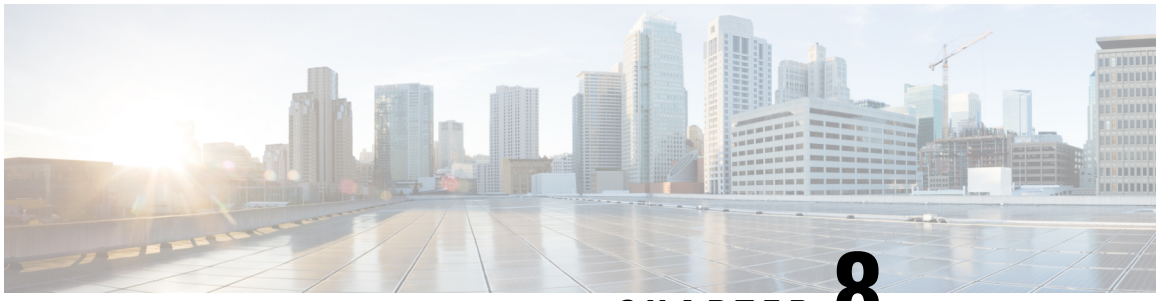
, a single server performs acceptably with up to 10,000 users.

CUNI Notifications only

, a single server performs acceptably with up to 6000 users. Each individual subscription is for 500 users or lesser.

Combined CUMI COMET and CUNI Notifications

, a single server performs acceptably with up to 5000 users. Each individual CUNI subscription is for 500 users or lesser.



CHAPTER 8

Cisco Unity Connection Messaging Interface (CUMI) API -- Dispatch Message Operations

About Dispatch Messages

A

Dispatch message

is a message that needs to go to one and only one member of a group. When the message is accepted by any one user, it is no longer available to other users. When the message is rejected by a user in the group, it is removed from the user's voicemail list.

-

Sending a Dispatch Message

Sending a Dispatch Message is very similar to sending any other CUMI Message. To sending a Dispatch message, a Boolean field (<Dispatch>) is set on the POST API call.

```
<Dispatch>true</Dispatch>
```

For more details on using the CUMI to send messages, refer here

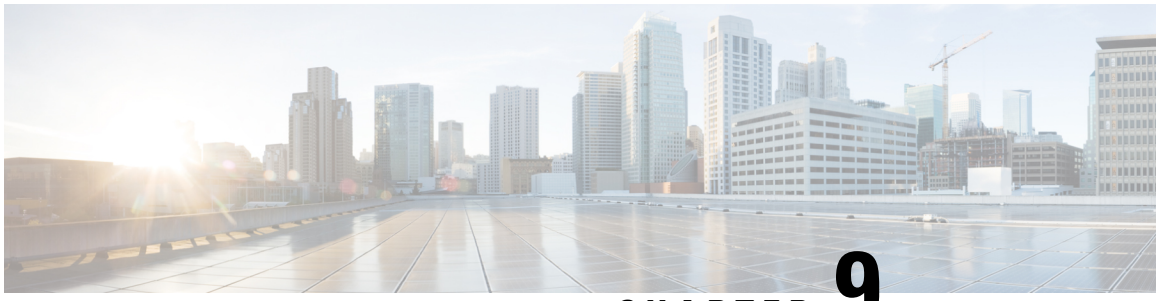
-

Accepting or Rejecting a Dispatch Message

A POST API call on the Message can be used to accept or reject a Dispatch message.

```
POST /vmrest/messages/{messageObjectId}?method=accept
POST /vmrest/messages/{messageObjectId}?method=reject
```

-



CHAPTER 9

Cisco Unity Connection Messaging Interface (CUMI) API -- Using the CUMI API for Broadcast Messages

About Broadcast Messages

Broadcast messages are system-owned messages that are meant to be heard by all users. The target user is either the signed-in user or the user specified by an administrator by using the optional "userobjectid=<user object id>" parameter.

URI	Method	Description
`mailbox/broadcastmessages`	GET	Returns a list of broadcast messages that are active and have not yet been heard by the user.
`mailbox/broadcastmessages/<message id>/voicefile`	GET	Returns the broadcast message voice file.
`mailbox/broadcastmessages/<message id>?read`	POST	Marks the broadcast message as heard by the target user.
`mailbox/broadcastmessages`	POST	Sends a broadcast message (see below).

-

Sending Broadcast Messages Reference

A broadcast message can be sent by a POST request to the URI noted in the table above. The content of the request is "mutipart/form-data". There must be two pieces of data in the request in the following order:

1. A broadcast message in either "application/xml" or "application/json" format.

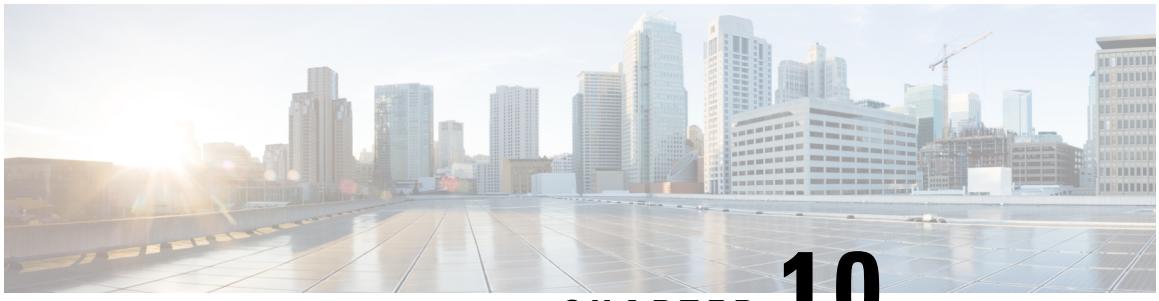
2. Audio data as "audio/wav."

The only user-modifiable fields for a broadcast message are the start date, end date, and the audio content. The rest of the fields are managed internally.

Schema

```
<xs:complexType name="BroadcastMessage">
 <xs:all minOccurs="0">
 <xs:element name="URI" type="xs:anyURI" minOccurs="0" />
 <xs:element name="ObjectId" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The primary key for this table. A globally unique, system-generated
identifier for a BroadcastMessage object.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="StreamFileObjectId" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation>StreamFileObjectId - use StreamFile instead of this column. The unique
 identifier of the StreamFile object
 containing the name of the WAV file that is the broadcast message.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="VoiceFileURI" type="xs:anyURI" minOccurs="0" />
 <xs:element name="SubscriberObjectId" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The unique identifier of the Subscriber object that sent this broadcast
 message.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="UserURI" type="xs:anyURI" minOccurs="0" />
 <xs:element name="CreationDate" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The date and time when the message was created.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="StartDate" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The date and time when the message becomes active.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="EndDate" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The date and time when the message expires.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="LastModificationSubscriberObjectId" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The unique identifier of the subscriber that last modified the
message.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="LastModificationUserURI" type="xs:anyURI" minOccurs="0" />
 <xs:element name="LastModificationDate" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation> The date and time the message was last modified.</xs:documentation>
 </xs:annotation>
 </xs:element>
 <xs:element name="StreamFile" type="xs:string" minOccurs="0">
 <xs:annotation>
 <xs:documentation>Name of audio file for this broadcast message. The name of the WAV file
 containing
 the recorded audio (voice name, greeting, etc.) for the parent object.</xs:documentation>

 </xs:annotation>
 </xs:element>
 </xs:all>
 </xs:complexType>
```



CHAPTER 10

Cisco Unity Connection Messaging Interface (CUMI) API -- Preventing Messages from Being Automatically Deleted (Investigative Hold)

Summary

Cisco Unity Connection provides two methods for automatically deleting messages from user mailboxes:

- Message aging determines when messages are automatically deleted for each user based on which message aging policy is assigned to the user.
- Message expiration automatically replaces .wav files with a generic "this message has expired" recording when the corresponding message has reached a specified age in days. This is a system-wide setting.

To create an archive of messages that you want to retain after they would otherwise be automatically deleted:

1. Create one or more new users whose mailboxes will be used for archiving messages. For example, you might create a second mailbox for every user whose messages you want to archive, or you might create a mailbox for an entire department.
2. Create a message-aging policy that does not age messages, and assign this policy to all of the mailboxes in which you want to archive messages past their automatic deletion date.
3. Use the CUMI API to forward the message to the applicable archive mailbox, and set the InvestigativeHold flag for the message to 1 by adding the following to the message resource:

```
<InvestigativeHold>1</InvestigativeHold>
```

Messages marked for investigative hold are not subject to message expiration.

Sample Perl Code for Forwarding a Message and Marking It for Investigative Hold

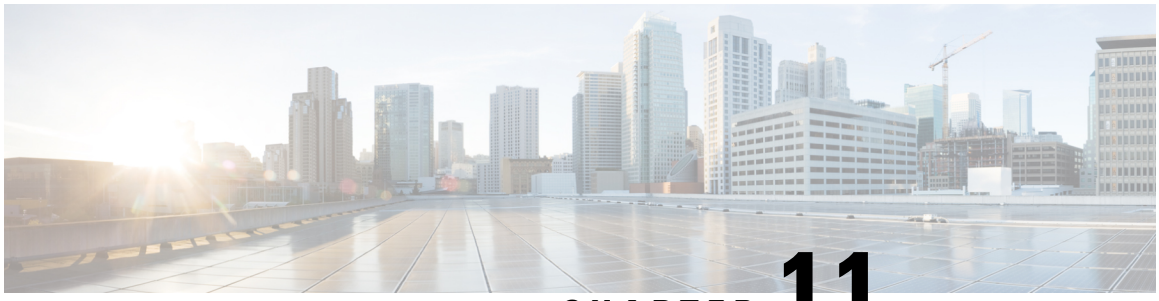
```
my $SERVER = 'cuc_server';
 my $PASSWORD = 'cuc_password';
 my $USER = 'cuc_user';
 sub ForwardMessage {
 my $recipient = shift(@_);
 my $messageObjectID = shift(@_);
 my $userObjectID = shift(@_);
 my $subject = shift(@_);
 my $credentials="$PASSWORD";
 my $url=https://$SERVER/vmrest/messages?messageid=$messageObjectID&
 userobjectid=$userObjectID;
 my $ua = LWP::UserAgent->new;
 my $header = HTTP::Headers->new;
 $header->header("Content-Type","multipart/mixed;boundary=Boundary");
 my $req = HTTP::Request->new(POST => $url, $header, $credentials);
 $req->authorization_basic($USER,$PASSWORD);
 #Set investigative hold on/off (1=on, 0=off)
 my $invHold = 1;
 my $part = <<END_OF_PART;
 --Boundary
 Content-Type: application/xml
 <?xml version="1.0" encoding="UTF-8" standalone="yes"?><Message>
 <InvestigativeHold>$invHold</InvestigativeHold><Subject>$subject</Subject>
 <ArrivalTime>0</ArrivalTime><FromSub>false</FromSub></Message>
 --Boundary
 Content-Type: application/xml
 <?xml version="1.0" encoding="UTF-8"
standalone="yes"?><Recipients><Recipient><Type>TO</Type>
 <Address><SmtpAddress>$recipient</SmtpAddress></Address></Recipient></Recipients>
 --Boundary--
 END_OF_PART
 print "\nForwarding message to ", $recipient, "\n";
 $req->content($part);
 my $response = $ua->request($req);
 };
```

-

Partial XML Message Schema

```
<xs:complexType name="Message">
 <xs:all>
 <\!\- fields that can be modified by the client at any time \->
 <xs:element name="Subject" type="xs:string" minOccurs="0" />
 <xs:element name="Read" type="xs:boolean" minOccurs="0" />
 <\!\- flags that can be set by the client on send \->
 <xs:element name="Dispatch" type="xs:boolean" minOccurs="0" />
 <xs:element name="Secure" type="xs:boolean" minOccurs="0" />
 <xs:element name="Priority" type="priorityType" minOccurs="0" />
 <xs:element name="Sensitivity" type="sensitivityType" minOccurs="0" />
 <xs:element name="ReadReceiptRequested" type="xs:boolean" minOccurs="0" />
 <xs:element name="DeliveryReceiptRequested" type="xs:boolean" minOccurs="0" />
 <xs:element name="InvestigativeHold" type="xs:boolean" minOccurs="0" />
```

-



CHAPTER 11

Cisco Unity Connection Messaging Interface (CUMI) API -- HTTP Samples

Send Message (Audio Data on Client)

Note that the audio data portion of the message is truncated after the first few bytes:

```
Hypertext Transfer Protocol
POST /vmrest/messages?userobjectid=b81bc5e4-e4c5-4743-9d84-34a401a5ba28 HTTP/1.1\r\n
Content-Type: multipart/form-data;boundary=Boundary_1_18607473_1256039585421\r\n
Accept: application/json\r\n
User-Agent: Java/1.6.0_11\r\n
Host: cuc-install-67.cisco.com\r\n
Connection: keep-alive\r\n
Authorization: Basic Y2NtYWRtaW5pc3RyYXRvcjplY3NidWxhYg==\r\n
Credentials: ccmadministrator:ecsbulab
Content-Length: 8001\r\n
[Content length: 8001]
\r\n
No. Time Source Destination Protocol Info
48 1.510814 10.93.225.254 10.93.230.219 HTTP Continuation or non-HTTP traffic
Frame 48 (1514 bytes on wire, 1514 bytes captured)
Ethernet II, Src: Ibm_71:6a:c6 (00:1a:64:71:6a:c6), Dst: Cisco_1e:d9:00 (00:23:04:1e:d9:00)

Internet Protocol, Src: 10.93.225.254 (10.93.225.254), Dst: 10.93.230.219 (10.93.230.219)

Transmission Control Protocol, Src Port: lumimgrd (4741), Dst Port: http (80), Seq: 905,
Ack: 2488, Len: 1460
Hypertext Transfer Protocol
\r\n
 Data (1458 bytes)
0000 2d 2d 42 6f 75 6e 64 61 72 79 5f 31 5f 31 38 36 --Boundary_1_186
0010 30 37 34 37 33 5f 31 32 35 36 30 33 39 35 38 35 07473_1256039585
0020 34 32 31 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 421..Content-Typ
0030 65 3a 20 61 70 70 6c 69 63 61 74 69 6f 6e 2f 6a e: application/j
0040 73 6f 6e 0d 0a 0d 0a 7b 22 53 75 62 6a 65 63 74 son....{"Subject
0050 22 3a 22 6d 75 6c 74 69 70 6c 65 20 72 65 63 69 ":"multiple reci
0060 70 69 65 6e 74 73 20 73 65 6e 64 20 6d 65 73 73 pients send mess
0070 61 67 65 20 74 65 73 74 22 2c 22 41 72 72 69 76 age test","Arriv
0080 61 6c 54 69 6d 65 22 3a 22 30 22 2c 22 46 72 6f alTime":"0","Fro
0090 6d 53 75 62 22 3a 22 66 61 6c 73 65 22 2c 22 46 mSub":"false","F
00a0 72 6f 6d 56 6d 49 6e 74 53 75 62 22 3a 22 66 61 romVmIntSub":"fa
00b0 6c 73 65 22 7d 0d 0a 2d 2d 42 6f 75 6e 64 61 72 lse"}..--Boundar
00c0 79 5f 31 5f 31 38 36 30 37 34 37 33 5f 31 32 35 y_1_18607473_125
00d0 36 30 33 39 35 38 35 34 32 31 0d 0a 43 6f 6e 74 6039585421..Cont
00e0 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69 63 ent-Type: applic
00f0 61 74 69 6f 6e 2f 6a 73 6f 6e 0d 0a 0d 0a 7b 22 ation/json....{"
0100 52 65 63 69 70 69 65 6e 74 22 3a 5b 7b 22 54 79 Recipient":[{"Ty
0110 70 65 22 3a 22 54 4f 22 2c 22 41 64 64 72 65 73 pe":"TO","Addres
0120 73 22 3a 7b 22 55 73 65 72 47 75 69 64 22 3a 22 s":{"UserGuid":"
0130 62 38 31 62 63 35 65 34 2d 65 34 63 35 2d 34 37 b81bc5e4-e4c5-47
0140 34 33 2d 39 64 38 34 2d 33 34 61 34 30 31 61 35 43-9d84-34a401a5
0150 62 61 32 38 22 7d 7d 2c 7b 22 54 79 70 65 22 3a ba28"}},{"Type":
0160 22 42 43 43 22 2c 22 41 64 64 72 65 73 73 22 3a "BCC","Address":
0170 7b 22 53 6d 74 70 41 64 64 72 65 73 73 22 3a 22 {"SmtpAddress":"
0180 6f 70 65 72 61 74 6f 72 40 63 75 63 2d 69 6e 73 operator@cuc-ins
0190 74 61 6c 6c 2d 36 37 2e 63 69 73 63 6f 2e 63 6f tall-67.cisco.co
01a0 6d 22 7d 7d 2c 7b 22 54 79 70 65 22 3a 22 43 43 m"}},{"Type":"CC
01b0 22 2c 22 41 64 64 72 65 73 73 22 3a 7b 22 4f 62 ","Address":{"Ob
01c0 6a 65 63 74 49 64 22 3a 22 31 61 62 61 38 64 39 jectId":"1aba8d9
01d0 39 2d 39 31 37 62 2d 34 31 35 38 2d 38 33 31 39 9-917b-4158-8319
01e0 2d 66 30 35 33 63 30 64 31 39 35 66 65 22 2c 22 -f053c0d195fe","
01f0 54 79 70 65 22 3a 22 44 49 53 54 52 49 42 55 54 Type":"DISTRIBUT
0200 49 4f 4e 4c 49 53 54 22 7d 7d 5d 7d 0d 0a 2d 2d IONLIST"}}]}..--
0210 42 6f 75 6e 64 61 72 79 5f 31 5f 31 38 36 30 37 Boundary_1_18607
0220 34 37 33 5f 31 32 35 36 30 33 39 35 38 35 34 32 473_125603958542
0230 31 0d 0a 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 1..Content-Type:
0240 20 61 75 64 69 6f 2f 77 61 76 0d 0a 0d 0a 52 49 audio/wav....RI
0250 46 46 c0 1c 00 00 57 41 56 45 66 6d 74 20 10 00 FF....WAVEfmt ..
0260 00 00 07 00 01 00 40 1f 00 00 40 1f 00 00 01 00 ......@...@.....
0270 08 00 66 61 63 74 04 00 00 00 8f 1c 00 00 64 61 ..fact........da
```

```
0280 74 61 90 1c 00 00 ff ff ff ff ff ff ff ff ff ff ta..............
0290 ff ff 7e 7e 7e 7e ff fe fd fd fd fe fe 7e 7e 7e ..22:49
```

Send Message (Audio Data Recorded Via CUTI on Server)

```
No. Time Source Destination Protocol Info
 61 9.984028 10.93.225.254 10.93.230.219 HTTP POST
/vmrest/messages?userobjectid=b81bc5e4-e4c5-4743-9d84-34a401a5ba28 HTTP/1.1
 Frame 61 (405 bytes on wire, 405 bytes captured)
 Ethernet II, Src: Ibm_71:6a:c6 (00:1a:64:71:6a:c6), Dst: Cisco_1e:d9:00 (00:23:04:1e:d9:00)

 Internet Protocol, Src: 10.93.225.254 (10.93.225.254), Dst: 10.93.230.219 (10.93.230.219)

 Transmission Control Protocol, Src Port: taurus-wh (1610), Dst Port: http (80), Seq: 2279,
 Ack: 4485, Len: 351
 Hypertext Transfer Protocol
 POST /vmrest/messages?userobjectid=b81bc5e4-e4c5-4743-9d84-34a401a5ba28 HTTP/1.1\r\n
 Content-Type: multipart/form-data;boundary=Boundary_1_28113457_1256068997078\r\n
 Accept: application/json\r\n
 User-Agent: Java/1.6.0_11\r\n
 Host: cuc-install-67.cisco.com\r\n
 Connection: keep-alive\r\n
 Authorization: Basic Y2NtYWRtaW5pc3RyYXRvcjplY3NidWxhYg==\r\n
 Credentials: ccmadministrator:ecsbulab
 Content-Length: 754\r\n
 [Content length: 754]
 \r\n
 0000 00 23 04 1e d9 00 00 1a 64 71 6a c6 08 00 45 00 .#......dqj...E.
 0010 01 87 5a a3 40 00 80 06 c1 39 0a 5d e1 fe 0a 5d ..Z.@....9.]...]
 0020 e6 db 06 4a 00 50 23 1f 44 2a 2b c4 79 ff 50 18 ...J.P#.D*+.y.P.
 0030 fb 26 df 0d 00 00 50 4f 53 54 20 2f 76 6d 72 65 .&....POST /vmrm
 0040 73 74 2f 6d 65 73 73 61 67 65 73 3f 75 73 65 72 st/messages?user
 0050 6f 62 6a 65 63 74 69 64 3d 62 38 31 62 63 35 65 objectid=b81bc5e
 0060 34 2d 65 34 63 35 2d 34 37 34 33 2d 39 64 38 34 4-e4c5-4743-9d84
 0070 2d 33 34 61 34 30 31 61 35 62 61 32 38 20 48 54 -34a401a5ba28 HT
 0080 54 50 2f 31 2e 31 0d 0a 43 6f 6e 74 65 6e 74 2d TP/1.1..Content-
 0090 54 79 70 65 3a 20 6d 75 6c 74 69 70 61 72 74 2f Type: multipart/
 00a0 66 6f 72 6d 2d 64 61 74 61 3b 62 6f 75 6e 64 61 form-data;bounda
 00b0 72 79 3d 42 6f 75 6e 64 61 72 79 5f 31 5f 32 38 ry=Boundary_1_28
 00c0 31 31 33 34 35 37 5f 31 32 35 36 30 36 38 39 39 113457_125606899
 00d0 37 30 37 38 0d 0a 41 63 63 65 70 74 3a 20 61 70 7078..Accept: ap
 00e0 70 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 0d 0a plication/json..
 00f0 55 73 65 72 2d 41 67 65 6e 74 3a 20 4a 61 76 61 User-Agent: Java
 0100 2f 31 2e 36 2e 30 5f 31 31 0d 0a 48 6f 73 74 3a /1.6.0_11..Host:
 0110 20 63 75 63 2d 69 6e 73 74 61 6c 6c 2d 36 37 2e cuc-install-67.
 0120 63 69 73 63 6f 2e 63 6f 6d 0d 0a 43 6f 6e 6e 65 cisco.com..Conne
 0130 63 74 69 6f 6e 3a 20 6b 65 65 70 2d 61 6c 69 76 ction: keep-aliv
 0140 65 0d 0a 41 75 74 68 6f 72 69 7a 61 74 69 6f 6e e..Authorization
 0150 3a 20 42 61 73 69 63 20 59 32 4e 74 59 57 52 74 : Basic Y2NtYWRt
 0160 61 57 35 70 63 33 52 79 59 58 52 76 63 6a 70 6c aW5pc3RyYXRvcjpl
 0170 59 33 4e 69 64 57 78 68 59 67 3d 3d 0d 0a 43 6f Y3NidWxhYg==..Co
 0180 6e 74 65 6e 74 2d 4c 65 6e 67 74 68 3a 20 37 35 ntent-Length: 75
 0190 34 0d 0a 0d 0a 4....
 No. Time Source Destination Protocol Info
 62 9.984053 10.93.225.254 10.93.230.219 HTTP Continuation or non-HTTP traffic
 Frame 62 (808 bytes on wire, 808 bytes captured)
 Ethernet II, Src: Ibm_71:6a:c6 (00:1a:64:71:6a:c6), Dst: Cisco_1e:d9:00 (00:23:04:1e:d9:00)

 Internet Protocol, Src: 10.93.225.254 (10.93.225.254), Dst: 10.93.230.219 (10.93.230.219)

 Transmission Control Protocol, Src Port: taurus-wh (1610), Dst Port: http (80), Seq: 2630,
 Ack: 4485, Len: 754
 Hypertext Transfer Protocol
 \r\n
 Data (752 bytes)
 Data: 2D2D426F756E646172795F315F32383131333435375F3132...
 0000 00 23 04 1e d9 00 00 1a 64 71 6a c6 08 00 45 00 .#......dqj...E.
 0010 03 1a 5a a4 40 00 80 06 bf a5 0a 5d e1 fe 0a 5d ..Z.@......]...]
 0020 e6 db 06 4a 00 50 23 1f 45 89 2b c4 79 ff 50 18 ...J.P#.E.+.y.P.
```

```
0030 fb 26 e0 a0 00 00 0d 0a 2d 2d 42 6f 75 6e 64 61 .&......--Bounda
0040 72 79 5f 31 5f 32 38 31 31 33 34 35 37 5f 31 32 ry_1_28113457_12
0050 35 36 30 36 38 39 39 37 30 37 38 0d 0a 43 6f 6e 56068997078..Con
0060 74 65 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69 tent-Type: appli
0070 63 61 74 69 6f 6e 2f 6a 73 6f 6e 0d 0a 0d 0a 7b cation/json....{
0080 22 53 75 62 6a 65 63 74 22 3a 22 73 65 6e 64 20 "Subject":"send
0090 6d 65 73 73 61 67 65 20 74 65 73 74 22 2c 22 41 message test","A
00a0 72 72 69 76 61 6c 54 69 6d 65 22 3a 22 30 22 2c rrivalTime":"0",
00b0 22 46 72 6f 6d 53 75 62 22 3a 22 66 61 6c 73 65 "FromSub":"false
00c0 22 2c 22 46 72 6f 6d 56 6d 49 6e 74 53 75 62 22 ","FromVmIntSub"
00d0 3a 22 66 61 6c 73 65 22 7d 0d 0a 2d 2d 42 6f 75 :"false"}..--Bou
00e0 6e 64 61 72 79 5f 31 5f 32 38 31 31 33 34 35 37 ndary_1_28113457
00f0 5f 31 32 35 36 30 36 38 39 39 37 30 37 38 0d 0a _1256068997078..
0100 43 6f 6e 74 65 6e 74 2d 54 79 70 65 3a 20 61 70 Content-Type: ap
0110 70 6c 69 63 61 74 69 6f 6e 2f 6a 73 6f 6e 0d 0a plication/json..
0120 0d 0a 7b 22 52 65 63 69 70 69 65 6e 74 22 3a 7b ..{"Recipient":{
0130 22 54 79 70 65 22 3a 22 54 4f 22 2c 22 41 64 64 "Type":"TO","Add
0140 72 65 73 73 22 3a 7b 22 55 73 65 72 47 75 69 64 ress":{"UserGuid
0150 22 3a 22 62 38 31 62 63 35 65 34 2d 65 34 63 35 ":"b81bc5e4-e4c5
0160 2d 34 37 34 33 2d 39 64 38 34 2d 33 34 61 34 30 -4743-9d84-34a40
0170 31 61 35 62 61 32 38 22 2c 22 44 69 73 70 6c 61 1a5ba28","Displa
0180 79 4e 61 6d 65 22 3a 22 4f 70 65 72 61 74 6f 72 yName":"Operator
0190 22 7d 7d 7d 0d 0a 2d 2d 42 6f 75 6e 64 61 72 79 "}}}..--Boundary
01a0 5f 31 5f 32 38 31 31 33 34 35 37 5f 31 32 35 36 _1_28113457_1256
01b0 30 36 38 39 39 37 30 37 38 0d 0a 43 6f 6e 74 65 068997078..Conte
01c0 6e 74 2d 54 79 70 65 3a 20 61 70 70 6c 69 63 61 nt-Type: applica
01d0 74 69 6f 6e 2f 78 6d 6c 0d 0a 0d 0a 3c 3f 78 6d tion/xml....<?xm
01e0 6c 20 76 65 72 73 69 6f 6e 3d 22 31 2e 30 22 20 l version="1.0"
01f0 65 6e 63 6f 64 69 6e 67 3d 22 55 54 46 2d 38 22 encoding="UTF-8"
0200 20 73 74 61 6e 64 61 6c 6f 6e 65 3d 22 79 65 73 standalone="yes
0210 22 3f 3e 3c 43 61 6c 6c 43 6f 6e 74 72 6f 6c 3e "?><CallControl>
0220 3c 6f 70 3e 50 4c 41 59 3c 2f 6f 70 3e 3c 72 65 <op>PLAY</op><re
0230 73 6f 75 72 63 65 54 79 70 65 3e 53 54 52 45 41 sourceType>STREA
0240 4d 3c 2f 72 65 73 6f 75 72 63 65 54 79 70 65 3e M</resourceType>
0250 3c 72 65 73 6f 75 72 63 65 49 64 3e 38 36 61 61 <resourceId>86aa
0260 66 31 65 32 2d 63 34 64 62 2d 34 65 30 38 2d 61 f1e2-c4db-4e08-a
0270 36 65 39 2d 38 32 65 31 37 30 36 61 37 39 66 34 6e9-82e1706a79f4
0280 2e 77 61 76 3c 2f 72 65 73 6f 75 72 63 65 49 64 .wav</resourceId
0290 3e 3c 6c 61 73 74 52 65 73 75 6c 74 3e 30 3c 2f ><lastResult>0</
02a0 6c 61 73 74 52 65 73 75 6c 74 3e 3c 73 70 65 65 lastResult><spee
02b0 64 3e 31 30 30 3c 2f 73 70 65 65 64 3e 3c 76 6f d>100</speed><vo
02c0 6c 75 6d 65 3e 31 30 30 3c 2f 76 6f 6c 75 6d 65 lume>100</volume
02d0 3e 3c 73 74 61 72 74 50 6f 73 69 74 69 6f 6e 3e ><startPosition>
02e0 30 3c 2f 73 74 61 72 74 50 6f 73 69 74 69 6f 6e 0</startPosition
02f0 3e 3c 2f 43 61 6c 6c 43 6f 6e 74 72 6f 6c 3e 0d ></CallControl>.
0300 0a 2d 2d 42 6f 75 6e 64 61 72 79 5f 31 5f 32 38 .--Boundary_1_28
0310 31 31 33 34 35 37 5f 31 32 35 36 30 36 38 39 39 113457_125606899
0320 37 30 37 38 2d 2d 0d 0a 7078--..
```

-