Back to: CUMI API Overview

Contents

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.

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. audio 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" />
<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

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>
```

Back to: CUMI API Overview