

**Links to Other API pages:** [Cisco Unity Connection APIs](#)

CUNI Guide Contents
<a href="#">API Overview</a>
<a href="#">CUNI Event Schema</a>
<b><a href="#">Subscribing to and Processing Notification Events</a></b>
<a href="#">CUNI FAQs</a>

## Contents

- [1 Subscribing to Notification Events](#)
- [2 Processing Notification Events](#)
  - ◆ [2.1 Example](#)

## Subscribing to Notification Events

Subscribing to notification events is done through the web service interface, and can be as simple as a single web service call to "subscribe".

At a minimum you should pass in:

- The callback URL where XML notifications will be posted by the Notifier
- The date/time when the subscription will expire

It is also possible to pass in a list of the resources (userid) that you are interested in receiving events for, although that can also be done via a subsequent call to "addResourceIdToSubscription". We recommend that you leave the eventTypeList empty, indicating that you would like to receive all types of message events.

This example shows making a call to subscribe in Java, using Axis2 stubs generated from the WSDL:

```
Calendar expiration = java.util.Calendar.getInstance();
com.cisco.unity.messageeventservice.MessageEventServiceStub.Subscribe s =
    new com.cisco.unity.messageeventservice.MessageEventServiceStub.Subscribe();

// Set subscription expiry to two months from now.
expiration.add(Calendar.MONTH, 2);
s.setExpiration(expiration);

// Specify the users we're interested in, which is just the operator for this example.
ArrayOfString resourceList = new ArrayOfString();
resourceList.addString("operator");

s.setResourceIdList(resourceList);
s.setExpiration(expiration);

// Set the callback information - a username and password can be specified, but they are optional.
MessageEventServiceStub.CallbackServiceInfo c = new MessageEventServiceStub.CallbackServiceInfo();
c.setCallbackServiceUrl(callbackUrl);
c.setPassword(callbackPassword);
c.setUsername(callbackUserId);
s.setCallbackServiceInfo(c);
```

```
// Subscribe!
SubscribeResponse r = stub.subscribe(s);
```

## Processing Notification Events

Events will be delivered via HTTP POST to the callback URL that was provided in the call to subscribe.

There are four basic types of events:

Event Type	Description
NEW_MESSAGE	Sent when a new message arrives in a user Inbox.
SAVED_MESSAGE	Sent when a message is marked as read.
UNREAD_MESSAGE	Sent when a message is marked as unread.
DELETED_MESSAGE	Sent when a message is deleted. (Note that this is sent on either a "hard" or "soft" delete. In the case of a soft delete, when the message is deleted from the deleted items list, no further event is sent out.)

## Example

This is an example of an event xml. Keep in mind that it is valid to have more than one messageInfo per messageEvent:

```
<?xml version="1.0" ?>
<messageEvent subscriptionId="00bdcfd3-159a-48d3-ac7b-2f3b4f83db6c"
  eventType="SAVED_MESSAGE"
  eventTime="11:15:40 PM 10/30/2008"
  mailboxId="abell"
  displayName="Alexander Bell"
  USN="2265">
  <messageInfo messageId="72204bd7-e5c3-446e-adb6-ae5f80db26fb"
    receiveTime="04:15:40 PM 10/30/2008"
    uid="543"
    msgType="Voice"
    priority="Normal-Priority"
    sender="null"
    callerAni="null"/>
</messageEvent>
```