**XMS Smoke Test Guide**

XMS-STS is a java application that is used to execute a series of basic XMS use cases to validate Powermedia XMS installation and configuration independent of the user’s application.

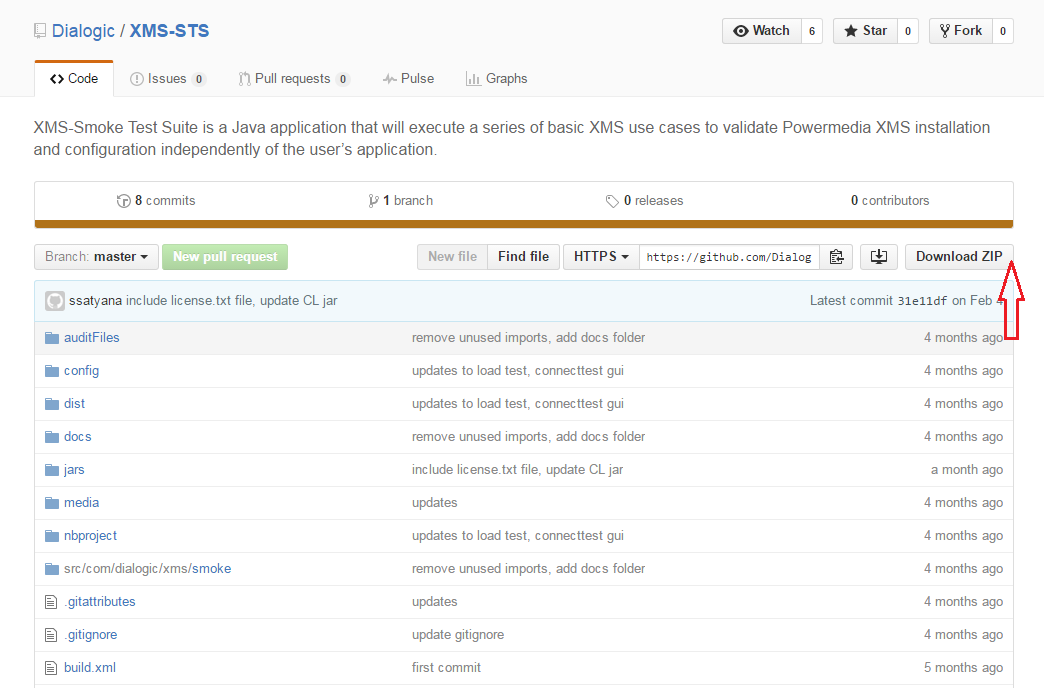
Currently, there are 7 use cases. These have to be executed manually by selecting and adding them to the run list. Each of the test cases generates an audit file consisting details related to the selected test and its status.

**Prerequisites**

* JAVA 1.8

**Obtaining the Project**

To use XMS-STS, download the zip folder from Github - <https://github.com/Dialogic/XMS-STS>



**Design**

XMS-STS consists of the following components/packages

Packages:

* gui – This consists of all the GUI components which allows the user to select the test to run and set the run configuration for the selected test case.
* test – This package consists of the main components shared by all the tests.
* examples – Consists of a list of use cases.
* stim – TODO: this will include stimulated test cases.

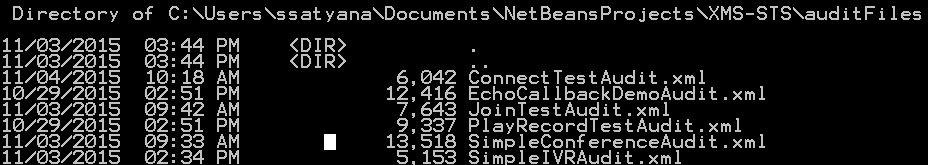
Components:

* Test Manager – This is the main program that runs the test case in sequential order from the run list.
* Test case – There are 7 test cases: - ConnectTest, EchoCallbackDemo, JoinTest, LoadTest, PlayRecordTest, SimpleConferenceTest, SimpleIVRTest. Each of these test cases will be explained in detail in the following sections.
* Controller – GUI which allows the user to select the test cases to be executed. This window also shows the progress and the status of the running tests.
* Configuration – Each test case has a configuration file used to execute the test. This configuration file consists of information like the mode of service, IP address, file name, no. of times to execute etc. This will be explained in detail in following sections
* Audit/Reporting – The execution of each test case will be recorded in the audit file. Checkpoints are used to track the status of the test case. This file includes the checkpoint information, info messages, configuration details, no. of steps passed/failed, test status (success/failure) etc.

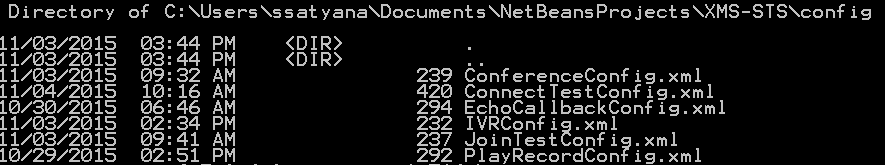
Reporting:

* Audit files generated after the test completion are stored in the auditFile directory under XMS-STS project.
* Config details entered by the user are stored in xml format in the config directory under XMS-STS project.

AuditFiles directory:



Config file directory:



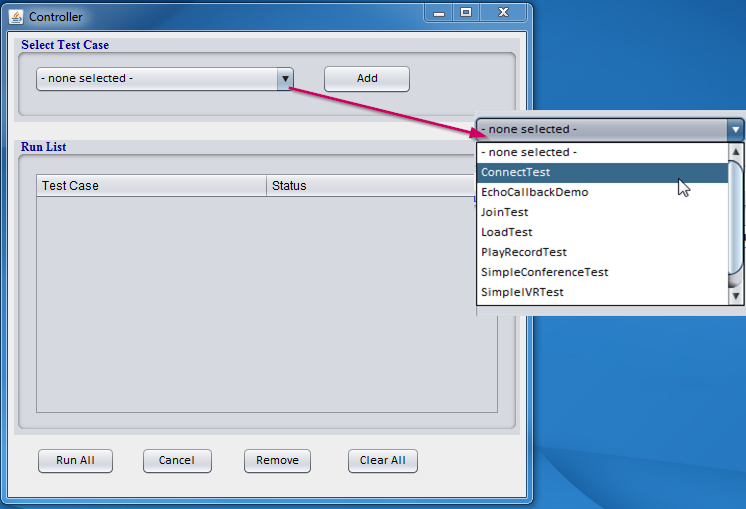
**Procedure**

Unzip the file downloaded from Github and then go to the top level directory where you will see the /dist directory. From the top level directory run the tool as follows:

C:\Users\ssatyana\Desktop\PROJECT-3\DocsPics\CmdPrompt.PNG

Following are the steps to run test case(s). The basic flow is explained using the ConnectTest use case.

**Step 1**: Running the jar will display the controller window used to select the test case to run.



The controller window consists of the following components:

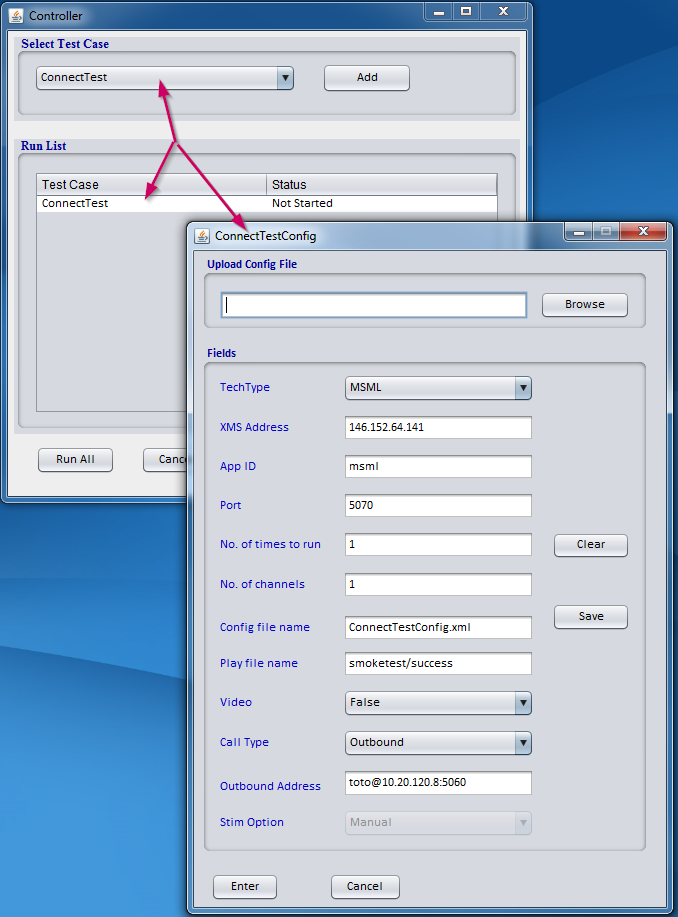
* Drop down menu – List of all the test cases
* Add button – After selecting the test case use this button to add to the run list
* Run list – This is a table showing the test case name and the status of the test case. The status keeps changing based on the current step of the test case
* Run All button – runs all the test cases in the run list
* Cancel – Closes the window and exits the program
* Remove – Removes a selected test from the run list
* Clear all button – Clears all the selected tests from the run list

**Step 2:** When a test case is selected (using the Add button) another window pops up. This window is used to set the config details for the selected test case.

*Note*:

* The config fields vary based on the test cases.
* When running the test cases for the first time, user has to enter the field details. Some of the fields will have placeholders to guide. Once the fields are entered, these get saved to the config files associated to each test case. When running the test for the second time, the fields will be auto-populated from the saved config files and used for further test. If the user changes the name of this config file, it will still auto-populate from the standard saved config file name.

For Example: Connect test has a default config file name “ConnectTestConfig.xml”. All the fields for Connect test will be saved to ConnectTestConfig.xml. If the user changes the config file name (i.e ConnectTestNew.xml), the next time the test is loaded, the default ConnectTestConfig.xml will be used to auto-populate the fields and not ConnectTestNew.xml.



Notice that on selecting the test case a config window pops up and the test is added to the run list.

The following are the components of the config window.

All the test cases will have the following common fields:

* Upload Config File – This is used to allow the user to upload configuration file from the system. The XML document format should match the sample provided at the end of this document.
* Tech Type (REST/MSML) – This allows the user to select the tech type used to run the selected test case.
* XMS Address – IP address of the XMS.
* App ID – This field is auto populated to match the tech type.
  + MSML – the user/ app id is “msml”.
  + REST – the user/app id is the REST application ID – “app”.
* Port – This field will be enabled only for MSML mode. This is the local port used to create the connector. If the user does not select one, default 5070 port will be selected.
* No. of times to run (count) – No. of times to run the selected test.
* No. of channels – No. of channels to run the selected test.
* Config file name – Standard file name used to store the entered config details.
* Stim Option(Manual/Auto) –Currently not available. Default is manual

Additional fields: Apart from the above common fields, there are additional fields that have to be set as per the test case.

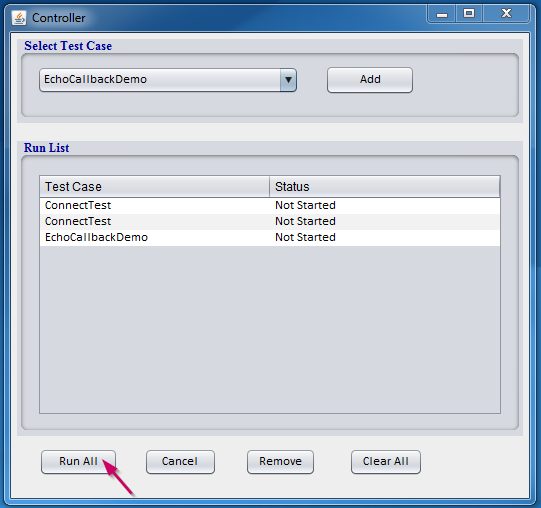
Connect Test:

* Play file name – file played on connecting to XMS. These are wav type files available under “XMS-STS/media” folder of the downloaded project. Place all these files under /var/lib/xms/media/en-US/smoketest (create smoketest directory) in XMS.
* Video (true/false) – Media type can be set to video or audio only. Default is audio.
* Call Type – Inbound or Outbound call
* Outbound address – Address to call to when the call type is outbound

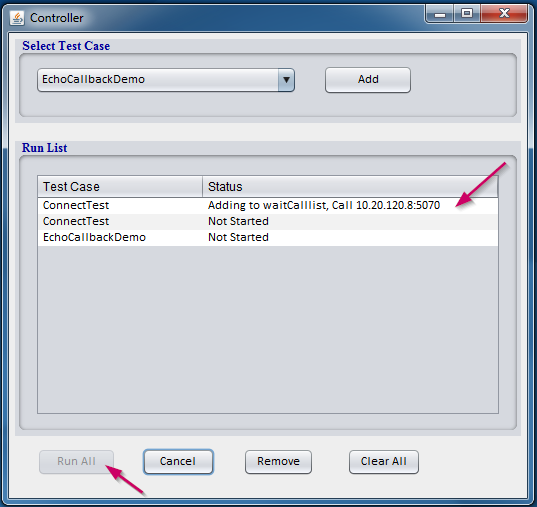
EchoCallbackDemo and PlayRecordTest:

* Record file name – file used to record
* Max time – Maximum time to record

**Step 3**: After entering the configuration details press “Enter”. This will take you back to the controller panel. Additional test cases can be selected. All tests will run sequentially.



Step 4: On clicking Run All, the tests will start executing. As the test progresses, status of the test can be viewed under the “Status” column.

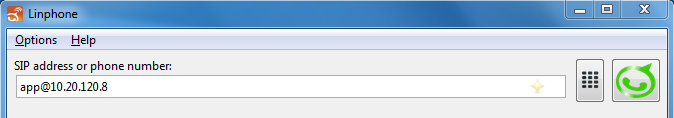


Now use any sip client (eg. Linphone ) to make a call to the application based on the tech type selected. The controller will display the address to call to in the status column. Use this address to call from linphone.

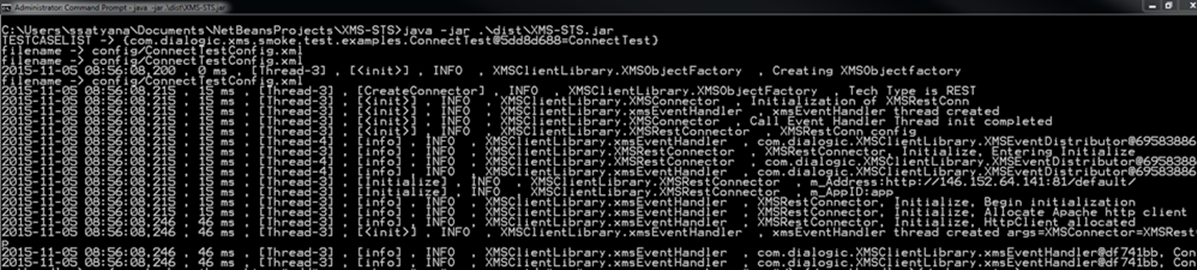
For MSML the calling address will be the local address where the application is running test@local\_ip:port.



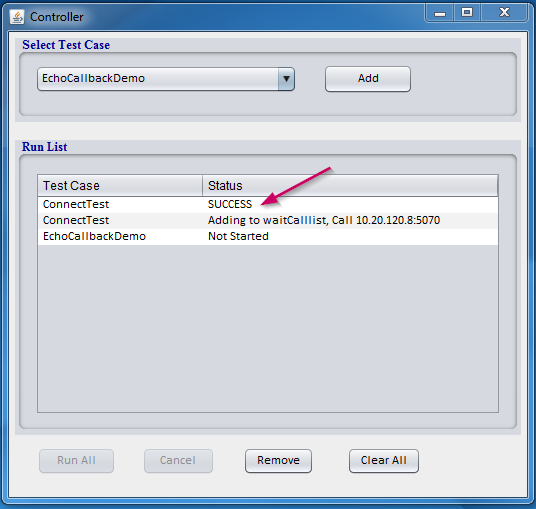
For REST the calling address will be app@xms\_ip\_addr



The call flow progress can also be tracked on the console window.



Once the test is completed the status displays if it was a SUCCESS/FAILURE.



**Test Case flow**

This section describes the flow/steps of each test case. The web sequence diagram shows an upper level understanding of the messages exchanged. For detailed information about the call connection and XML content, please refer the wireshark provided under “XMS-STS/docs” folder.

***Prompts used in the test cases are present in the “XMS-STS/media” directory in the zip file downloaded from Github. Store these .wav files under /var/lib/xms/media/en-US/smoketest (create smoketest folder) directory while testing. This path is hard coded in most of the test cases. If the file location is changed, please update the test cases to point to the correct directory.***

*Note: Web Sequence diagrams attached in this document will also be placed in the “XMS-STS/docs/SequenceDiagram” folder in the XMS-STS zip.*

**ConnectTest**

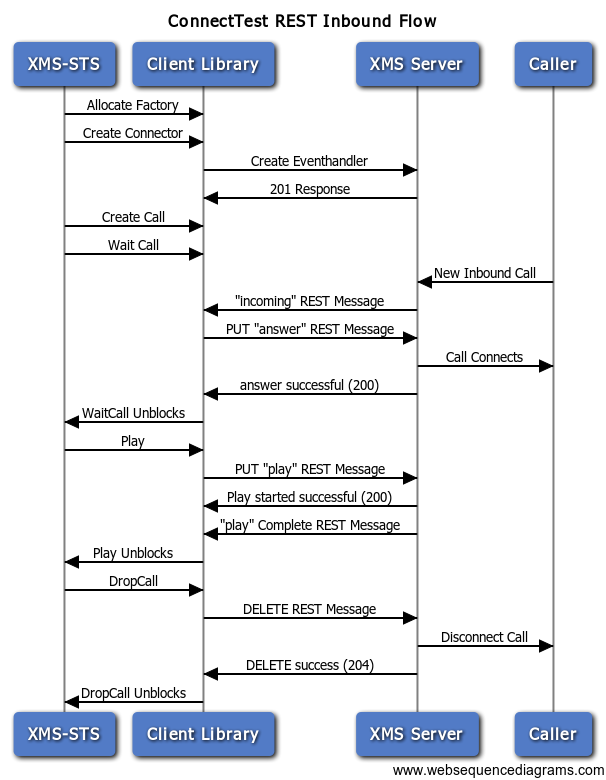
XMS-STS uses the Client Library to make an inbound or outbound call based on the configuration details entered by the user.

Following are the steps:

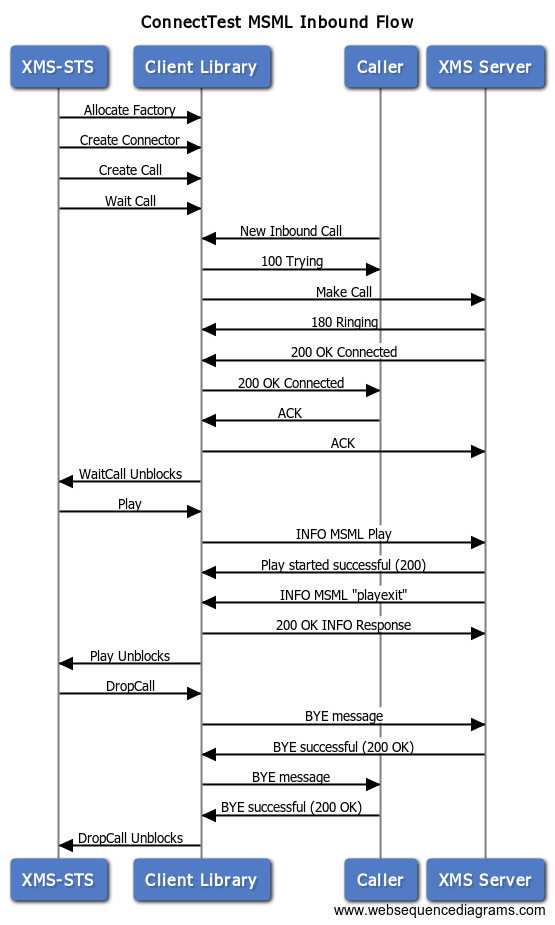
* 1. Make call (inbound/outbound). For inbound call use a sip phone to connect. For outbound call, the application uses the outbound address provided by the user in the config file.
  2. After the call is connected, a prompt is played “If you can hear this, you are successfully connected to XMS”
  3. Either the user can hang up the call or the application disconnects the call after 10s.

User can use this to make REST inbound/outbound call or MSML inbound/outbound call.

*Inbound REST:*

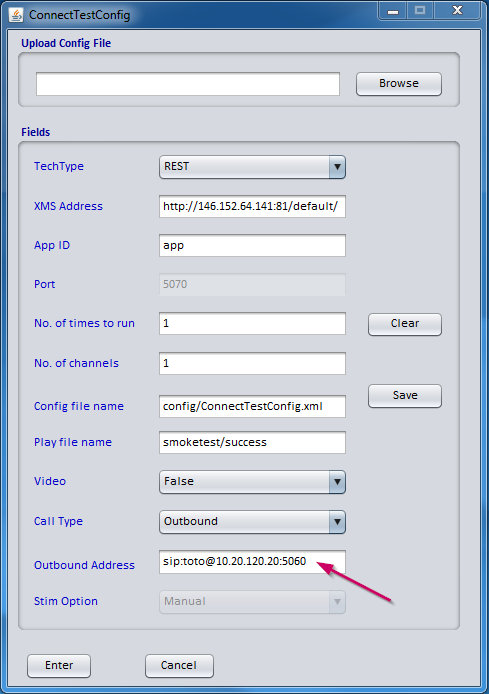


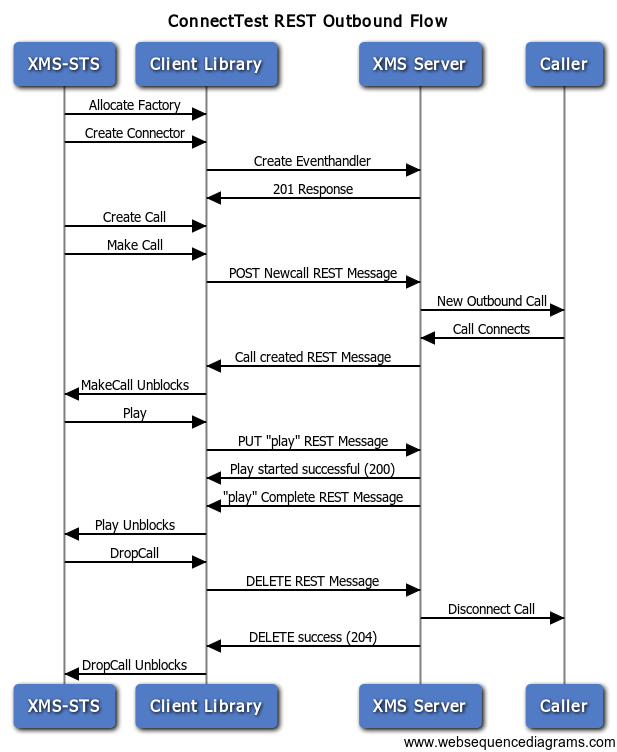
*Inbound MSML:*



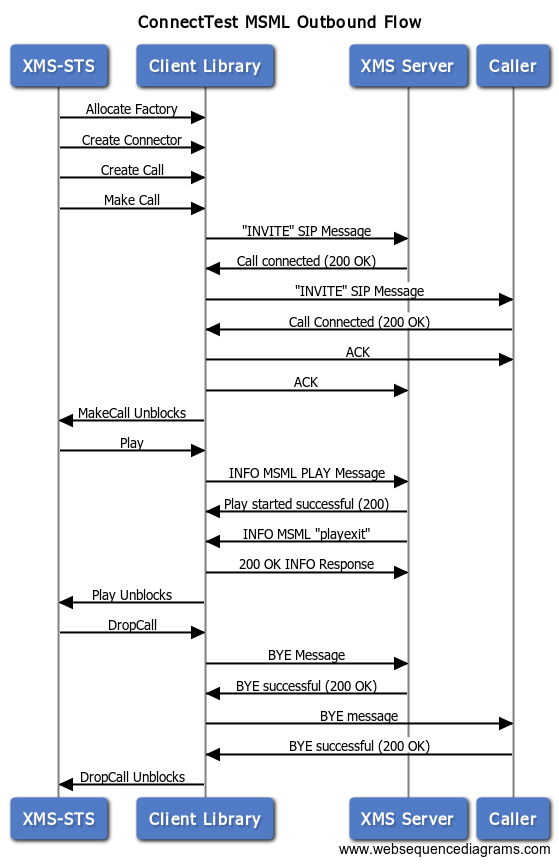
*Outbound REST:*

In the outbound test case, the address to call to should be mentioned in the outbound address field provided in the configuration (see below)





*Outbound MSML:*

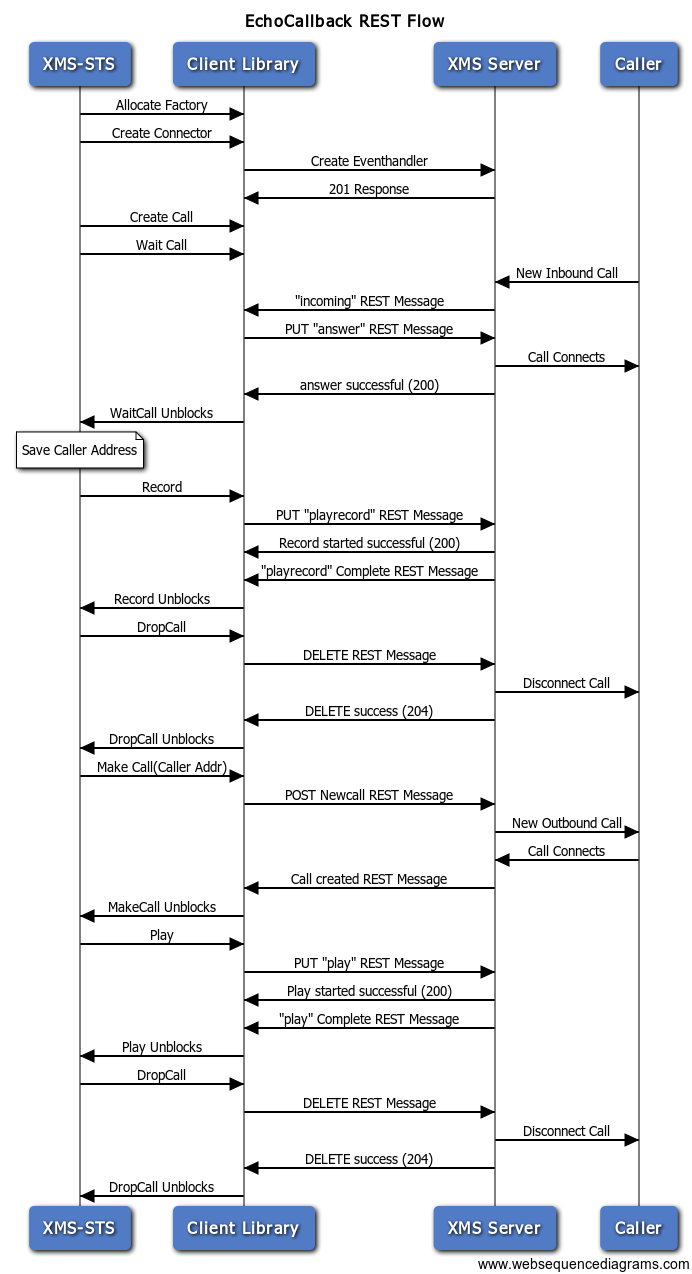


**EchoCallbackDemo**

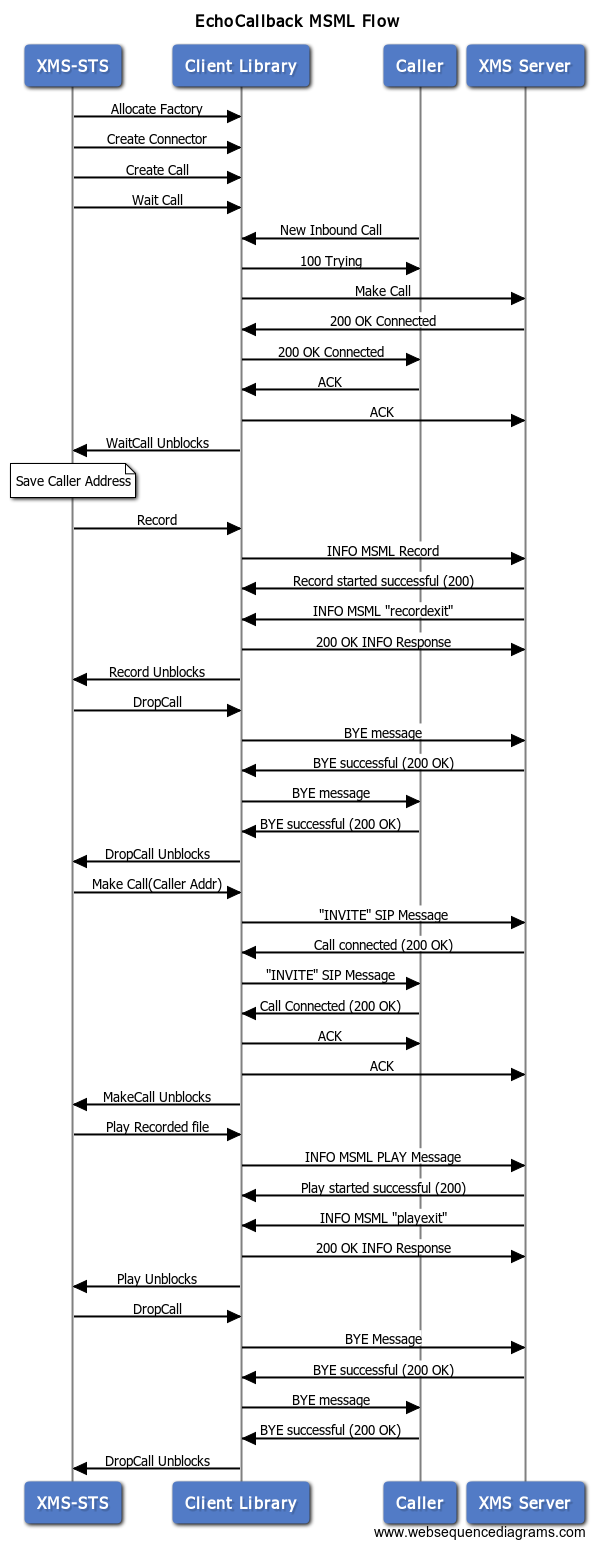
Steps:

1. Client (sip phone) calls the application (MSML mode) or XMS(REST mode).
2. Recording starts once the call is connected. Media is recorded for the max time set by the user in the config window, default is 10s.
3. Once recording is done, call is disconnected.
4. Application makes an outbound call to the same client (sip phone).
5. Plays back the recorded file and disconnects the call once the play completes.

*REST:*



*MSML:*



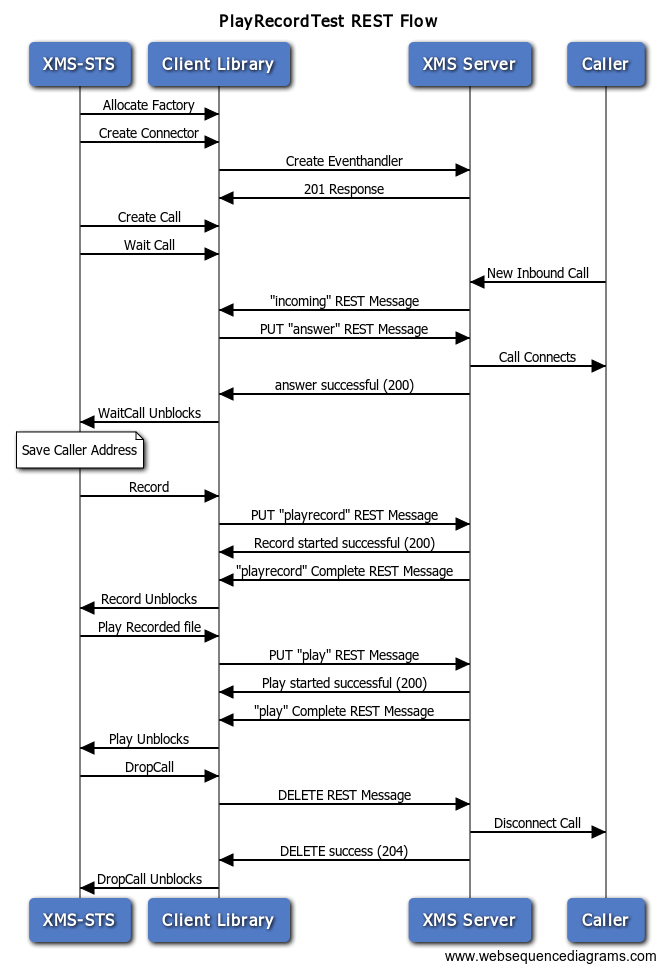
**PlayRecordTest**

This test is similar to EchoCallbackDemo test except there is no call back to the client.

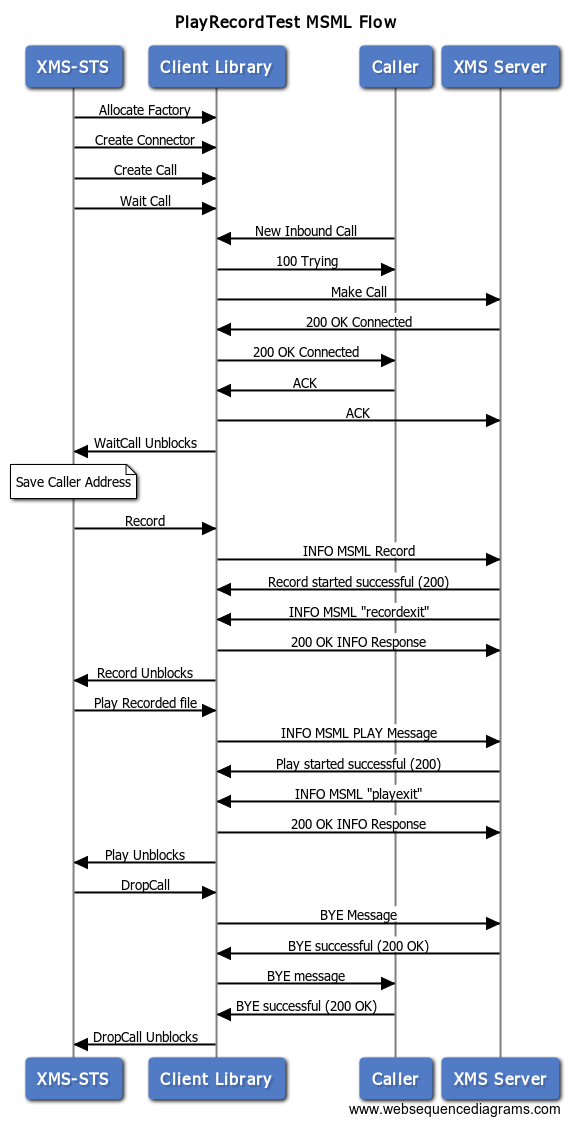
Steps:

1. Client calls into application.
2. Application starts recording after the call gets connected. Record time is based on the max time in the configuration settings.
3. Once recording is done, plays back the recorded file.
4. Disconnects the call.

*REST:*



*MSML:*



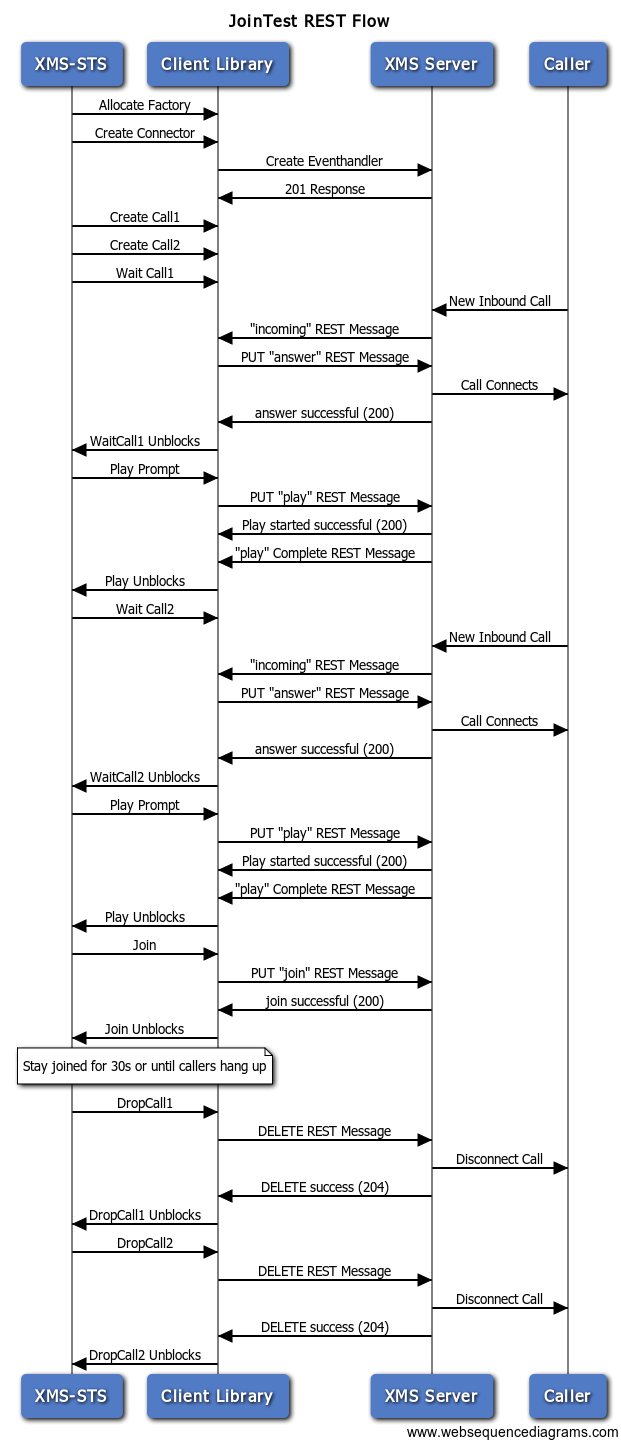
**JoinTest**

This test bridges two calls.

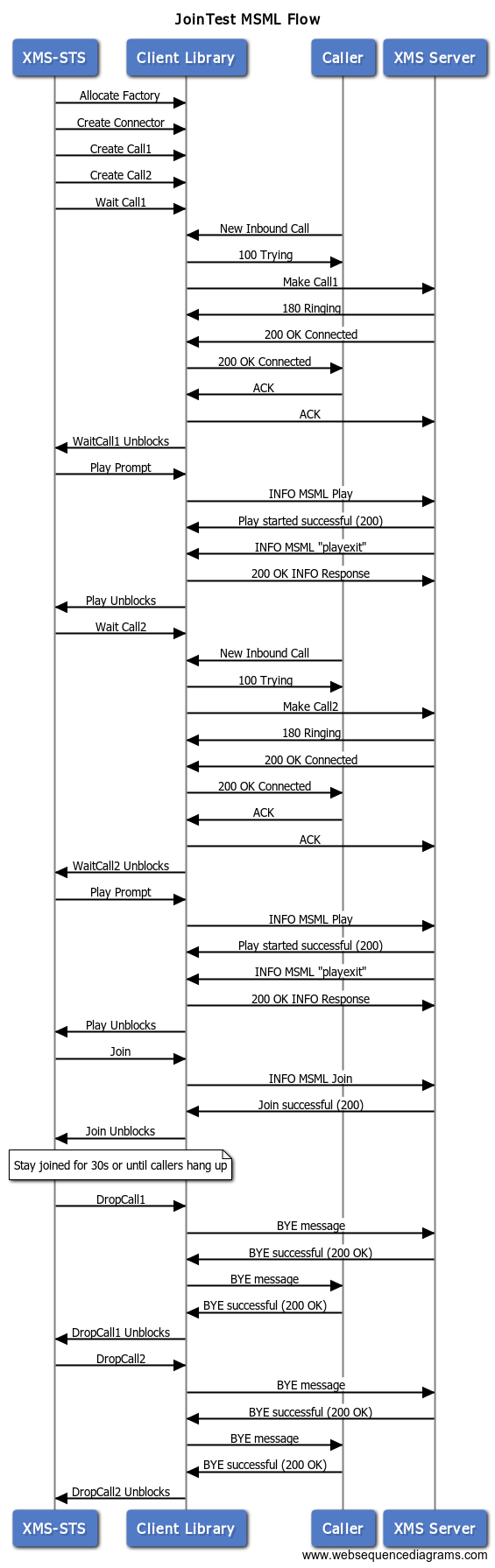
Steps:

1. Caller A dials into the application.
2. Caller A hears a basic prompt “Please wait for second caller to join”.
3. Caller B dials into the application.
4. Caller B hears a basic prompt “Now joining with first caller”.
5. Application joins caller A and caller B.
6. After the calls are joined, app waits for 30s or until one of the callers hang up and terminate calls

*REST:*



MSML:



**LoadTest**

Test runs asynchronously and requires call back functionality to handle the events.

Steps:

1. This is a load test, so user can place x number of calls into the application
2. On connecting, each call receives a prompt
3. Once the prompt finishes, the call is terminated

**SimpleConferenceTest**

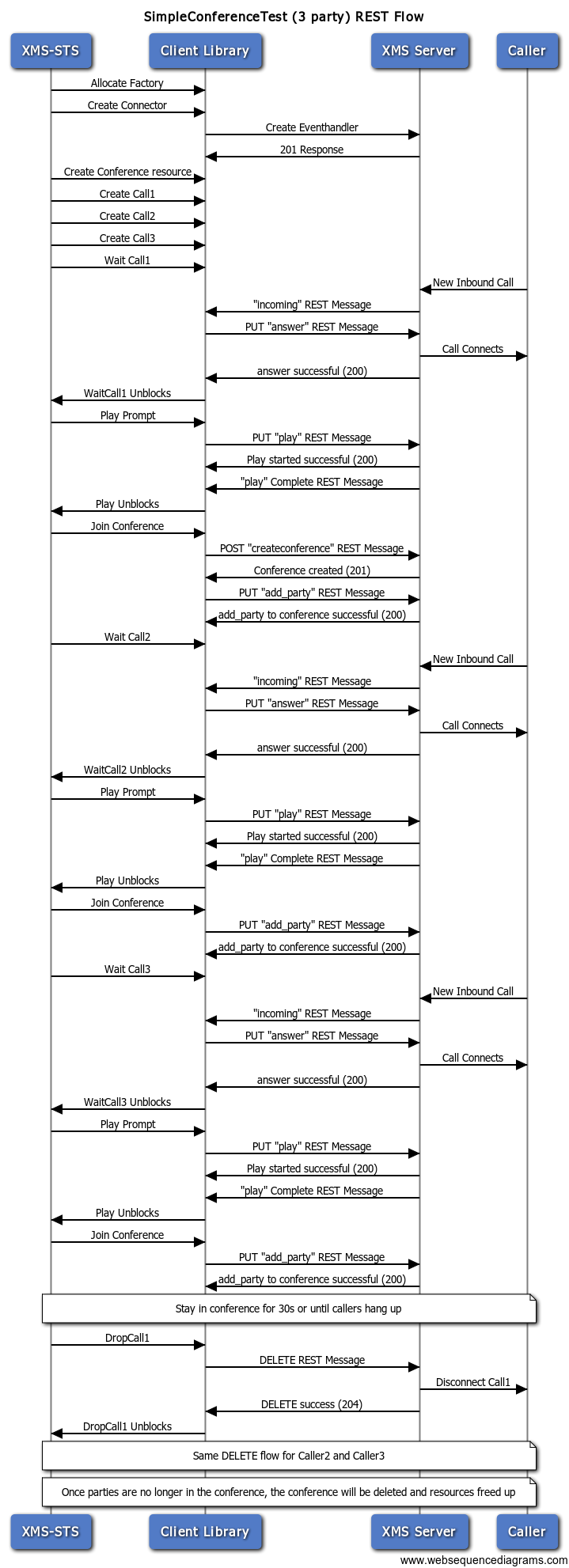
This is a basic 3 party conference.

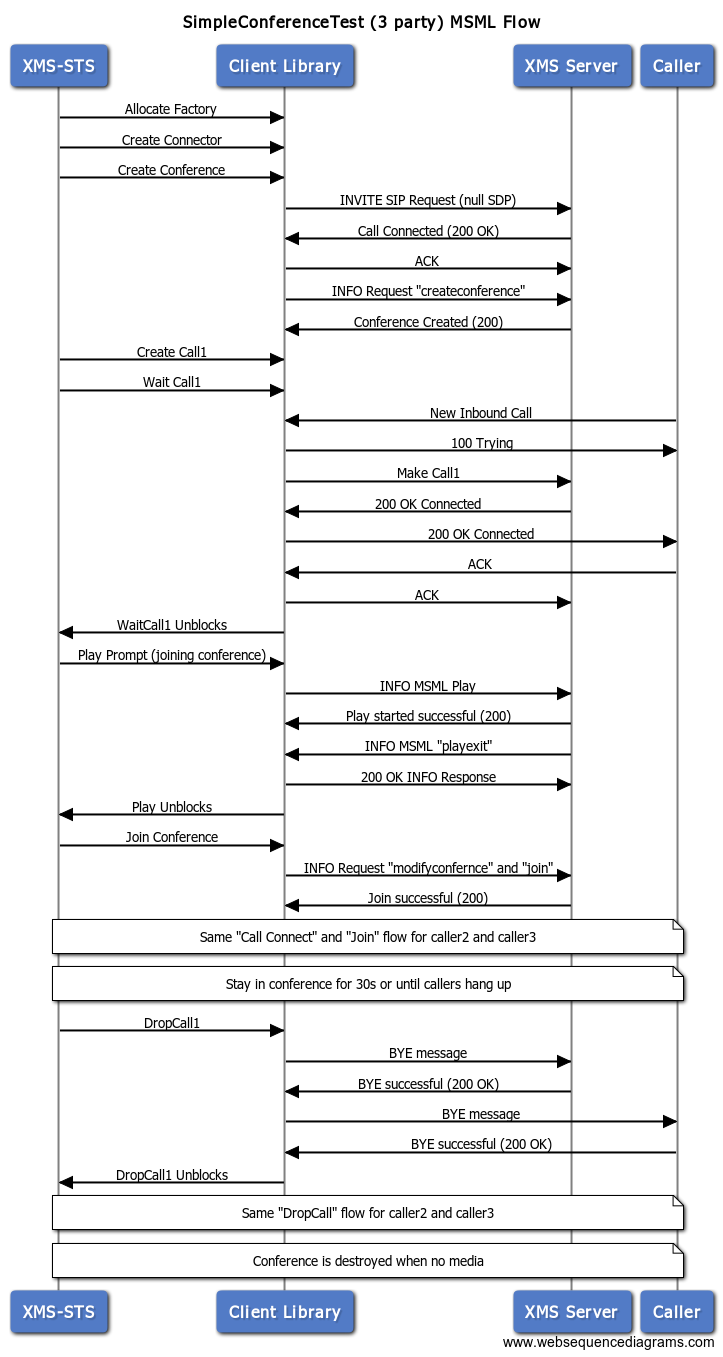
*REST* Steps:

1. Start the connector and is waiting for the callers to dial in
2. When the first caller dials in, a prompt is played "adding to conf" while the conference is established and the caller is added to the conference.
3. As additional callers dial in, the same prompt is played and the caller is added to the conference.
4. As conferees leave, the call leg parameters are cleared.
5. When the last conferee leaves, the conference is deleted.

*MSML* Steps:

1. Conference is created and is waiting for the callers to dial in
2. When the first caller dials in, a prompt is played "adding to conf" while the caller is added to the conference.
3. As additional callers dial in, the same prompt is played and the caller is added to the conference.
4. As conferees leave, the call leg parameters are cleared.
5. When the last conferee leaves, the conference is deleted.





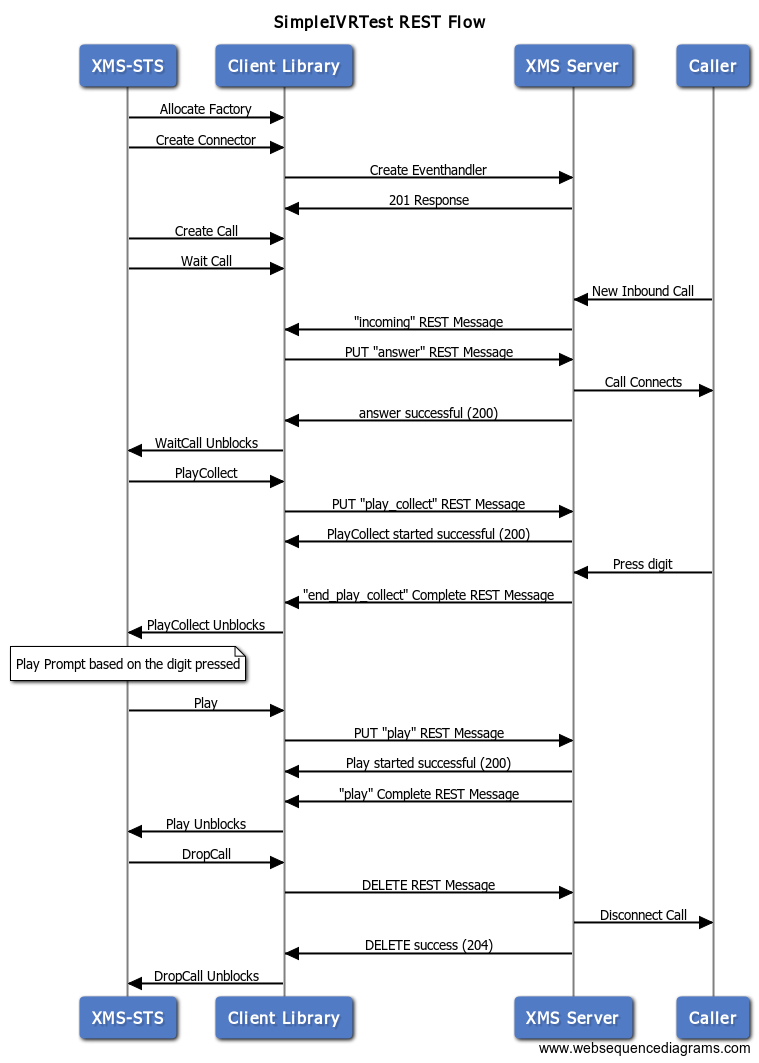
**SimpleIVRTest**

Test related to digit detection.

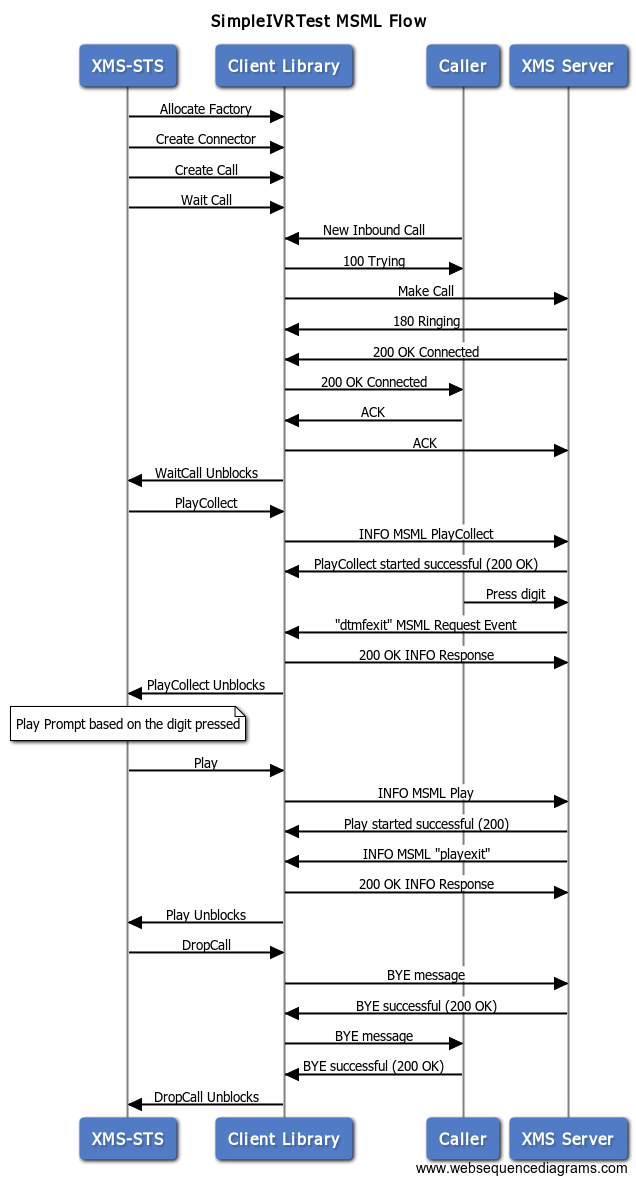
Steps:

1. Client calls into app.
2. As the caller gets connected, the application prompts to enter DTMF - "Press 1 for Sales, Press 2 for services, Press 3 for CEO"
3. Based on DTMF entered by the user, the next prompt is played using user’s selection - "you've reached \_\_\_\_" (selection)"

*REST:*



MSML:



**Reporting**

Audit files are created at the end of each test case execution for references. These audit files are maintained in “auditFiles” folder under XMS-STS project.

An audit file consists of details of the program execution and is recorded as checkpoints. They also have information about the configuration settings, success/failure of each checkpoint, short description about the checkpoint, long description with REST events or MSML content and finally the status of the test (SUCCESS/FAILURE).

While the test is executing, each of these checkpoints are also displayed on the controller window under the “status” column.

ConnectTestAudit Sample

<?xml version="1.0" encoding="UTF-8"?>

<java version="1.8.0\_40" class="java.beans.XMLDecoder">

<object class="com.dialogic.xms.smoke.test.Audit">

<void property="auditTestName">

<string>OutboundPlayTest</string>

</void>

<void property="checkpointCount">

<int>3</int>

</void>

<void property="configContents">

<object class="com.dialogic.xms.smoke.test.Config">

<void property="appID">

<string>app</string>

</void>

<void property="callType">

<string>Outbound</string>

</void>

<void property="channels">

<int>1</int>

</void>

<void property="count">

<int>1</int>

</void>

<void property="ipAddress">

<string>http://146.152.64.141:81/default/</string>

</void>

<void property="outboundAddress">

<string>sip:toto@10.20.120.20:5060</string>

</void>

<void property="playFileName">

<string>smoketest/success</string>

</void>

<void property="port">

<int>5070</int>

</void>

<void property="stim">

<string>Manual</string>

</void>

<void property="type">

<string>REST</string>

</void>

</object>

</void>

<void property="configFileName">

<string>config/ConnectTestConfig.xml</string>

</void>

<void property="numPass">

<int>2</int>

</void>

<void property="startTime">

<string>[10:49:22.826] </string>

</void>

<void property="testCheckpoints">

<object class="java.util.ArrayList">

<void method="add">

<object class="com.dialogic.xms.smoke.test.Checkpoint">

<void property="checkpointName">

<string>MakeCall</string>

</void>

<void property="checkpointStartTime">

<string>[10:49:23.450] </string>

</void>

<void property="checkpointStatus">

<string>SUCCESS</string>

</void>

<void property="longDesc">

<string>XMSRestEvt- ID=0befb42d-3cb2-41c2-acd5-c4f0692f6c3b Type=connected ResourceType=call event=&lt;xml-fragment type=&quot;connected&quot; resource\_id=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot; resource\_type=&quot;call&quot;&gt;&#13;

&lt;event\_data name=&quot;call\_id&quot; value=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot;/&gt;&#13;

&lt;event\_data name=&quot;called\_uri&quot; value=&quot;&amp;lt;sip:toto@10.20.120.20:5060&gt;;tag=Xia4S92&quot;/&gt;&#13;

&lt;event\_data name=&quot;caller\_uri&quot; value=&quot;&amp;lt;sip:146.152.64.141&gt;;tag=f0bce880-8d409892-13c4-50022-1639f-5af7ca0a-1639f&quot;/&gt;&#13;

&lt;event\_data name=&quot;media&quot; value=&quot;audio&quot;/&gt;&#13;

&lt;event\_data name=&quot;reason&quot; value=&quot;unknown&quot;/&gt;&#13;

&lt;event\_data name=&quot;type&quot; value=&quot;CONNECTED&quot;/&gt;&#13;

&lt;/xml-fragment&gt;</string>

</void>

<void property="shortDesc">

<string>Making an outbound call</string>

</void>

</object>

</void>

<void method="add">

<object class="com.dialogic.xms.smoke.test.Checkpoint">

<void property="checkpointName">

<string>Play</string>

</void>

<void property="checkpointStartTime">

<string>[10:49:24.445] </string>

</void>

<void property="checkpointStatus">

<string>SUCCESS</string>

</void>

<void property="longDesc">

<string>XMSRestEvt- ID=0befb42d-3cb2-41c2-acd5-c4f0692f6c3b Type=end\_play ResourceType=call event=&lt;xml-fragment type=&quot;end\_play&quot; resource\_id=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot; resource\_type=&quot;call&quot;&gt;&#13;

&lt;event\_data name=&quot;duration&quot; value=&quot;4470&quot;/&gt;&#13;

&lt;event\_data name=&quot;id&quot; value=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot;/&gt;&#13;

&lt;event\_data name=&quot;media\_id&quot; value=&quot;a7974944-1781-4839-aa92-e92131a7df08&quot;/&gt;&#13;

&lt;event\_data name=&quot;reason&quot; value=&quot;end&quot;/&gt;&#13;

&lt;event\_data name=&quot;status&quot; value=&quot;0 No Error&quot;/&gt;&#13;

&lt;event\_data name=&quot;transaction\_id&quot; value=&quot;a7974944-1781-4839-aa92-e92131a7df08&quot;/&gt;&#13;

&lt;event\_data name=&quot;type&quot; value=&quot;END\_PLAY&quot;/&gt;&#13;

&lt;/xml-fragment&gt;</string>

</void>

<void property="shortDesc">

<string>Playing</string>

</void>

</object>

</void>

<void method="add">

<object class="com.dialogic.xms.smoke.test.Checkpoint">

<void property="checkpointName">

<string>DropCall</string>

</void>

<void property="checkpointStartTime">

<string>[10:49:39.137] </string>

</void>

<void property="checkpointStatus">

<string>SUCCESS</string>

</void>

<void property="longDesc">

<string>XMSRestEvt- ID=0befb42d-3cb2-41c2-acd5-c4f0692f6c3b Type=end\_play ResourceType=call event=&lt;xml-fragment type=&quot;end\_play&quot; resource\_id=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot; resource\_type=&quot;call&quot;&gt;&#13;

&lt;event\_data name=&quot;duration&quot; value=&quot;4470&quot;/&gt;&#13;

&lt;event\_data name=&quot;id&quot; value=&quot;0befb42d-3cb2-41c2-acd5-c4f0692f6c3b&quot;/&gt;&#13;

&lt;event\_data name=&quot;media\_id&quot; value=&quot;a7974944-1781-4839-aa92-e92131a7df08&quot;/&gt;&#13;

&lt;event\_data name=&quot;reason&quot; value=&quot;end&quot;/&gt;&#13;

&lt;event\_data name=&quot;status&quot; value=&quot;0 No Error&quot;/&gt;&#13;

&lt;event\_data name=&quot;transaction\_id&quot; value=&quot;a7974944-1781-4839-aa92-e92131a7df08&quot;/&gt;&#13;

&lt;event\_data name=&quot;type&quot; value=&quot;END\_PLAY&quot;/&gt;&#13;

&lt;/xml-fragment&gt;</string>

</void>

<void property="shortDesc">

<string>Drop call</string>

</void>

</object>

</void>

</object>

</void>

<void property="testStatus">

<string>SUCCESS</string>

</void>

</object>

</java>

**Sample XML document for configuration setting**

Config files are also present in the “config” folder under XMS-STS project.

ConnectTest Sample

<xmsconfig>

<techtype>REST</techtype>

<baseurl>http://146.152.64.141:81/default/</baseurl>

<appid>app</appid>

<port>5070</port>

<count>1</count>

<channels>1</channels>

<configFile>config/ConnectTestConfig.xml</configFile>

<playFile>smoketest/success</playFile>

<video>False</video>

<callType>Outbound</callType>

<outboundAddress>sip:toto@10.20.120.20:5060</outboundAddress>

<stim>Manual</stim>

</xmsconfig>