MODIFICATION RECORD

Version 0.3    April 29, 2009
• Revised Procedure for all tests.
• Revised Observable Results for all tests

Version 0.2    January 20, 2009
• Removed Test 1.6 and Test 1.11.

Version 0.1    March 20, 2008
• Initial Draft.
ACKNOWLEDGMENTS

The University of New Hampshire InterOperability Laboratory would like to acknowledge the efforts of the following individuals in the development of this test suite.

<table>
<thead>
<tr>
<th>Name</th>
<th>Institution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Niels Widger</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>James Swan</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>Timothy Winters</td>
<td>University of New Hampshire</td>
</tr>
</tbody>
</table>
INTRODUCTION

Overview

The University of New Hampshire’s InterOperability Laboratory (UNH-IOL) is an institution designed to improve the interoperability of standards based products by providing an environment where a product can be tested against other implementations of a standard.

This interoperability test suite has been developed to help implementers verify that their products interoperate with other SIP implementations on the key features of some widely implemented extensions to the SIP specification. Failure to properly interoperate can lead to serious issues when deployed in a multi-vendor environment using popular SIP extensions.

Successful completion of all tests in this test suite does not guarantee that the device under test is compliant with the appropriate specification or that it will interoperate in all environments or scenarios. However, successful completion of these tests should provide a reasonable level of confidence that the device under test will function well in most multi-vendor environments.

Abbreviations and Acronyms

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>B2BUA</td>
<td>Back-to-Back User Agent</td>
</tr>
<tr>
<td>DNS</td>
<td>Domain Name System</td>
</tr>
<tr>
<td>DUT</td>
<td>Device Under Test</td>
</tr>
<tr>
<td>FQDN</td>
<td>Fully Qualified Domain Name</td>
</tr>
<tr>
<td>PS1</td>
<td>Proxy Server 1</td>
</tr>
<tr>
<td>PS2</td>
<td>Proxy Server 2</td>
</tr>
<tr>
<td>MOHS1</td>
<td>Music On Hold Server 1</td>
</tr>
<tr>
<td>CPS1</td>
<td>Call Park Server 1</td>
</tr>
<tr>
<td>RTP</td>
<td>Real-time Transport Protocol</td>
</tr>
<tr>
<td>SDP</td>
<td>Session Description Protocol</td>
</tr>
<tr>
<td>SIP</td>
<td>Session Initiation Protocol</td>
</tr>
<tr>
<td>UA</td>
<td>User Agent</td>
</tr>
<tr>
<td>UA1</td>
<td>User Agent 1</td>
</tr>
<tr>
<td>UA2</td>
<td>User Agent 2</td>
</tr>
<tr>
<td>UA3</td>
<td>User Agent 3</td>
</tr>
<tr>
<td>UA4</td>
<td>User Agent 4</td>
</tr>
<tr>
<td>UAC</td>
<td>User Agent Client</td>
</tr>
<tr>
<td>UAS</td>
<td>User Agent Server</td>
</tr>
</tbody>
</table>
Terms and Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Address-of-Record (AOR)</td>
<td>A SIP or SIPS URI pointing to a domain with a location service, which can map the URI to another URI where the user might be available. Typically, the location service is populated through registrations. An AOR is frequently thought of as the “public address” of the user.</td>
</tr>
<tr>
<td>Answerer</td>
<td>An agent which receives a session description from another agent describing aspects of desired media communication, and then responds to that with its own session description.</td>
</tr>
<tr>
<td>Call</td>
<td>An informal term that refers to some communication between peers, generally set up for the purposes of a multimedia conversation.</td>
</tr>
<tr>
<td>Dialog</td>
<td>A peer-to-peer SIP relationship between two UAs that persists for some time. A dialog is established by SIP messages such as a 2xx response to an INVITE request. A call identifier, local tag, and a remote tag identify a dialog. A dialog was formerly known as a call leg in RFC2543.</td>
</tr>
<tr>
<td>Final Response</td>
<td>A response that terminates a SIP transaction, as opposed to a provisional response that does not. All 2xx, 3xx, 4xx, 5xx and 6xx responses are final.</td>
</tr>
<tr>
<td>Header Field</td>
<td>A header field is a component of the SIP message header. A header field can appear as one or more header field rows. Header field rows consist of a header field name and zero or more header field values. Multiple header field values on a given header field row are separated by commas. Some header fields can only have a single header field value, and as a result, always appear as a single header field row.</td>
</tr>
<tr>
<td>Initiator, Calling Party, Caller</td>
<td>The party initiating a session (and dialog) with an INVITE request. A caller retains this role from the time that it sends the initial INVITE that established a dialog until the termination of that dialog.</td>
</tr>
<tr>
<td>Invitation</td>
<td>An INVITE request.</td>
</tr>
<tr>
<td>Invitee, Invited User, Called Party, Callee</td>
<td>The party that receives an INVITE request for the purpose of establishing a new session. A callee retains this role from the time it receives the INVITE until the termination of the dialog established by the request.</td>
</tr>
<tr>
<td>Multimedia Stream</td>
<td>A single media instance, e.g., an audio stream or a video stream as well as a single whiteboard or shared application group. In SDP, a media stream is described by an “m=” line and its associated attributes.</td>
</tr>
<tr>
<td>Offerer</td>
<td>An agent that generates a session description in order to create or modify a session.</td>
</tr>
<tr>
<td>Outbound Proxy</td>
<td>A proxy that receives requests from a client, even though it may not be the server resolved by the Request-URI. Typically, a UA is manually configured with an outbound proxy, or can learn about one through auto-configuration protocols.</td>
</tr>
<tr>
<td>Provisional Response</td>
<td>A response used by the server to indicate progress, but that does not terminate a SIP transaction. 1xx responses are provisional, other responses are considered final.</td>
</tr>
<tr>
<td><strong>Proxy Server</strong></td>
<td>An intermediate entity that acts as both a server and a client for the purposes of making requests on behalf of other clients. A proxy server primarily plays the role of routing, which means its job is to ensure that a request is sent to another entity “closer” to the targeted user. Proxies are also useful for enforcing policy (for example, making sure a user is allowed to make a call). A proxy interprets, and, if necessary, rewrites specific parts of a request message before forwarding it.</td>
</tr>
<tr>
<td><strong>Registrar</strong></td>
<td>A server that accepts REGISTER requests and places the information it receives in those requests into the location service for the domain it handles.</td>
</tr>
<tr>
<td><strong>Request</strong></td>
<td>A SIP message sent from a client to a server, for the purpose of invoking a particular operation.</td>
</tr>
<tr>
<td><strong>Response</strong></td>
<td>A SIP message sent from a server to a client, for indicating the status of a request sent from the client to the server.</td>
</tr>
<tr>
<td><strong>Session</strong></td>
<td>A multimedia session is a set of multimedia senders and receivers and the data streams flowing from senders to receivers. A multimedia conference is an example of a multimedia session. A session as defined for SDP can comprise one or more RTP sessions. As defined, a callee can be invited several times, by different calls, to the same session. If SDP is used, a session is defined by the concatenation of the SDP user name, session id, network type, address type, and address elements in the original field.</td>
</tr>
<tr>
<td><strong>Timer A</strong></td>
<td>The INVITE request retransmission interval when using the UDP transport layer. Initially this timer is set to the T1 timer interval (500ms default).</td>
</tr>
</tbody>
</table>
TEST ORGANIZATION

This document organizes tests by Section based on related test methodology or goals. Each group begins with a brief set of comments pertaining to all tests within that group. This is followed by a series of description blocks; each block describes a single test. The format of the description block is as follows:

**Test Label:**
The test label and title comprise the first line of the test block. The test label is composed by concatenating the short test suite name, the section number, the group number, and the test number within the group. These elements are separated by periods. The Test Number is the section, group and test number, also separated by periods.

**Purpose:**
The Purpose is a short statement describing what the test attempts to achieve. It is usually phrased as a simple assertion of the feature or capability to be tested.

**References:**
The References section lists cross-references to the specifications and documentation that might be helpful in understanding and evaluating the test and results.

**Node Requirements:**
The Node Requirements section is a short description of the features necessary of a particular node in the testing process.

**Discussion:**
The Discussion is a general discussion of the test and relevant section of the specification, including any assumptions made in the design or implementation of the test as well as known limitations.

**Test Setup:**
The Test Setup section describes the configuration of all devices prior to the start of the test. Different parts of the procedure may involve configuration steps that deviate from what is given in the test setup. If a value is not provided for a protocol parameter, then the protocol’s default is used for that parameter.

**Procedure:**
This section of the test description contains the step-by-step instructions for carrying out the test. These steps include such things as enabling interfaces, unplugging devices from the network, or sending packets from a test station. The test procedure also cues the tester to make observations, which are interpreted in accordance with the observable results given for that test part.

**Observable Results:**
This section lists observable results that can be examined by the tester to verify that the DUT is operating properly. When multiple observable results are possible, this section provides a short discussion on how to interpret them. The determination of a pass or fail for each test is usually based on how the DUT’s behavior compares to the results described in this section.

**Possible Problems:**
This section contains a description of known issues with the test procedure, which may affect test results in certain situations.
REFERENCES

The following documents are referenced in this text:


<table>
<thead>
<tr>
<th>Section</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>ACKNOWLEDGMENTS</td>
<td>3</td>
</tr>
<tr>
<td>INTRODUCTION</td>
<td>4</td>
</tr>
<tr>
<td>OVERVIEW</td>
<td>4</td>
</tr>
<tr>
<td>ABBREVIATIONS AND ACRONYMS</td>
<td>4</td>
</tr>
<tr>
<td>TERMS AND DEFINITIONS</td>
<td>5</td>
</tr>
<tr>
<td>TEST ORGANIZATION</td>
<td>7</td>
</tr>
<tr>
<td>REFERENCES</td>
<td>8</td>
</tr>
<tr>
<td>TABLE OF CONTENTS</td>
<td>9</td>
</tr>
<tr>
<td>General Node Requirements</td>
<td>10</td>
</tr>
<tr>
<td>Common Test Setup 1</td>
<td>11</td>
</tr>
<tr>
<td>Group 1:</td>
<td>14</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.1: CALL HOLD</td>
<td>15</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.2: CONSULTATION HOLD</td>
<td>16</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.3: MUSIC ON HOLD</td>
<td>18</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.4: TRANSFER - UNATTENDED</td>
<td>20</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.5: TRANSFER - ATTENDED</td>
<td>21</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.6: CALL FORWARDING UNCONDITIONAL</td>
<td>23</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.7: CALL FORWARDING - BUSY</td>
<td>24</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.8: CALL FORWARDING – NO ANSWER</td>
<td>25</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.9: 3-WAY CONFERENCE – THIRD PARTY IS ADDED</td>
<td>26</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.10: FIND-ME</td>
<td>27</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.11: CALL MANAGEMENT (INCOMING CALL SCREENING)</td>
<td>29</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.12: CALL MANAGEMENT (OUTGOING CALL SCREENING)</td>
<td>30</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.13: CALL PARKING</td>
<td>31</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.14: PARKED CALL PICKUP</td>
<td>32</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.15: AUTOMATIC REDIAL</td>
<td>33</td>
</tr>
<tr>
<td>TEST SIPFEAT.1.16: CLICK TO DIAL</td>
<td>34</td>
</tr>
</tbody>
</table>
General Node Requirements

- **SIP User Agent**
  - Ability to configure an outgoing proxy server.
  - Ability to put a call on hold.
  - Ability to take a call off hold.
  - Ability to park a call on a call park server.
  - Ability to transfer a call.
  - Ability to hold a 3-way conference.
  - Support for DNS A-Record Resolution.
  - Note: Many tests contained in this document can be run using IPv4 addresses.

- **SIP Proxy Server**
  - Ability to accept anonymous registrations.
  - Ability to perform digest authentication for user agent registration.
  - Ability to authenticate incoming and outgoing requests.
  - Ability to fork an incoming request using a list of possible contacts.
  - Support for DNS A-Record Resolution.
  - Note: Many tests contained in this document can be run using IPv4 addresses.

- **SIP Announcement Server**
  - Ability to play an announcement upon receiving an incoming call.

- **SIP Call Park Server**
  - Allow UA’s to park calls on the server and retrieve them at a later point.

- **SIP Music On Hold Server**
  - Ability to play music for user while on hold.
Common Test Setup 1

Common Configuration Parameters for UA1
- Register to PS1.
- FQDN is "ua1.ioltest.net"
- SIP URI is "sip:ua1@ioltest.net"

Common Configuration Parameters for UA2
- No registration.
- FQDN is "ua2.ioltest.net"
- SIP URI is "sip:ua2@ioltest.net"

Common Configuration Parameters for UA3
- Register to PS1.
- FQDN is "ua3.ioltest.net"
- SIP URI is "sip:ua3@ioltest.net"

Common Configuration Parameters for UA4
- Register to PS1.
- FQDN is "ua4.ioltest.net"
- SIP URI is "sip:ua4@ioltest.net"
Common Configuration Parameters for PS1
- FQDN is “ps1.ioltest.net”
- Accept registrations from UA1, UA2, UA3, and UA4.

Common Configuration Parameters for MOHS1
- FQDN is “mohs1.ioltest.net”

Common Configuration Parameters for AS1
- FQDN is “ms1.ioltest.net”

Common Configuration Parameters for CPS1
- FQDN is “cps1.ioltest.net”
Group 1:

Scope:

Tests in this group verify that the target devices are able to engage in SIP sessions that include the use of various features such as call hold, call parking, etc. This group of test applies only to User Agent Clients (UACs).

Overview:

Session Initiation Protocol (SIP), an application-layer control (signaling) protocol for creating, modifying, and terminating sessions with one or more participants. These sessions include Internet telephone calls, multimedia distribution, and multimedia conferences. SIP is extendible, allowing new features to be added to the base specification. These new features expand SIP’s capabilities to allow SIP devices to perform more complex call features. These tests aim to test a device’s ability to perform the most widely used of these new features.
Test SIPFEAT.1.1: Call Hold

**Purpose:** This test verifies that a UAC can successfully place a call on hold and then take it off hold with a UAS.

**References:**
- [SIP-SPEC]
- [SIP-DIAL] – Section 5.2
- [SDP-SPEC] – Section 6

**Node Requirements:**
See General Node Requirements.

**Discussion:** When a UAC wants to place a call on hold, it sends a new INVITE request to the UAS. The INVITE request indicates to the UAS that it will not be rendering any further audio received from the UAS. When the UAC eventually takes the call off hold, it sends another INVITE request, which indicates it will once again render audio coming from the UAS.

**Test Setup:**
Common Test Setup 1

**Procedure:**

**Part A:**
1. UA2 calls UA1.
2. UA1 answers the call from UA2.
3. UA1 places the call with UA2 on hold.
4. UA1 takes the call with UA2 off hold.
5. Put UA2 back on the hook.
6. Observe traffic on all networks.

**Observable Results:**

- **Part A**
  - **Step 2:** There must be audio flowing in both directions between UA1 and UA2.
  - **Step 3:** UA1 must place the call with UA2 on hold. There must not be audio flowing in either direction between UA1 and UA2.
  - **Step 4:** UA1 must take the call with UA2 off hold. There must be audio flowing in both directions between UA1 and UA2.
  - **Step 5:** There must not be audio flowing in either direction between UA1 and UA2.

**Possible Problems:**

- None
Test SIPFEAT.1.2: Consultation Hold

Purpose: This test verifies that a UAS can successfully place an existing call on hold, place and terminate a call with a third party, and then take the initial call off hold and terminate it.

References:
- [SIP-SPEC]
- [SIP-DIAL] – Section 5.2
- [SDP-SPEC] – Section 6

Node Requirements:
See General Node Requirements.

Discussion: A UAC that has just placed an existing call on hold may desire to call another UAS before returning to the original call. In this situation, the UAC first places the call on hold by sending a new INVITE request indicating one-way audio, then sends an INVITE request to create a new situation with the other UAS. After that session has been terminated, the UAC can take the original call off hold by sending an INVITE request indicating two-way audio between the UAC and UAS.

Test Setup:
Common Test Setup 1

Procedure:

Part A:
1. UA2 calls UA1.
2. UA1 answers the call from UA2.
3. UA1 places the call with UA2 on hold.
4. UA1 calls UA3.
5. UA3 answers the call from UA1.
6. UA1 terminates the call with UA3.
7. UA1 takes the call with UA2 off hold.
8. Put UA1 back on the hook.
9. Observe traffic on all networks.

Observable Results:
- Part A
  - Step 1: UA2 must call UA1.
  - Step 2: There must be audio flowing in both directions between UA1 and UA2.
  - Step 3: UA1 must place the call with UA2 on hold. There must not be audio flowing in either direction between UA1 and UA2.
  - Step 4: UA1 must call UA3.
  - Step 5: There must be audio flowing in both directions between UA1 and UA3.
  - Step 6: UA1 must terminate the call with UA3.
  - Step 7: UA1 must take the call with UA2 off hold. There must be audio flowing in both directions.
Possible Problems:

- None
Test SIPFEAT.1.3: Music On Hold

Purpose: This test verifies that a UAS can successfully place a call on hold by referring the other party to a music on hold server, and then resume the call by taking it off hold.

References:

- [SIP-SPEC]
- [SIP-REP]
- [SIP-REF]
- [SIP-UAC]
- [SIP-DIAL] – Section 5.2
- [SDP-SPEC] – Section 6

Node Requirements:
See General Node Requirements.

Discussion: When putting a UAS on hold, a UAC may desire that the UAS hears music while waiting on hold. This can be accomplished by first placing the call on hold, then instructing a Media Server to call the UAS, replacing the existing session between the UAC and UAS. This can be accomplished by sending a REFER request from the UAC to the Media Server, which will then send an INVITE request which includes a Replaces header to the UAS. At some later point, the UAC can resume the call with the UAS by sending an INVITE request with a Replaces header to the UAS.

Test Setup: Common Test Setup 1

Procedure:

Part A:
1. Configure UA1 to use MOHS1 as its MOHS server.
2. UA2 calls UA1.
3. UA1 answers the call from UA2.
4. UA1 places the call with UA2 on hold.
5. UA1 takes the call with UA2 off hold.
7. Observe traffic on all networks.

Observable Results:

- Part A
  - **Step 2:** UA1 must call UA2.
  - **Step 3:** There must be audio flowing in both directions between UA1 and UA2.
  - **Step 4:** UA1 must place the call with UA2 on hold using MOHS1 as its music on hold server. There must not be audio flowing in either direction between UA1 and UA2. There must be audio flowing from MOHS1 to UA2.
  - **Step 5:** UA1 must take the call with UA2 off hold. There must be audio flowing in both directions between UA1 and UA2. There must not be audio flowing from MOHS1 to UA2.
Possible Problems:

- None
Test SIPFEAT.1.4: Transfer - Unattended

Purpose: This test verifies that a UAC can perform an unattended transfer of a call with a UAS.

References:

- [SIP-SPEC]
- [SIP-REF]
- [SIP-CT]
- [SDP-SPEC]

Node Requirements:
See General Node Requirements.

Discussion: When a UAC wishes to transfer an existing call to another user, it can accomplish this by sending a REFER request containing the SIP-URI of the user in its Refer-To header to the UAS. After receiving the REFER request, the UAS will then end the session with the UAC and following that send a new INVITE request to the SIP-URI contained in the Refer-To header of the REFER request sent by the UAC.

Test Setup: Common Test Setup 1

Procedure:

Part A:

1. UA1 calls UA2.
2. UA2 answers the call.
3. UA1 transfers the call with UA2 to UA3.
4. Put UA1 back on hook.
5. Observe traffic on all networks.

Observable Results:

- Part A
  Step 1: UA1 must call UA2.
  Step 2: There must be audio flowing in both directions between UA1 and UA2.
  Step 3: UA1 must transfer the call with UA2 to UA3. There must not be audio flowing in either direction between UA1 and UA2. There must be audio flowing in both directions between UA2 and UA3.

Possible Problems:

- None
Test SIPFEAT.1.5: Transfer - Attended

**Purpose:** This test verifies that a UAC can successfully perform an attended transfer with a UAS.

**References:**

- [SIP-SPEC]
- [SIP-REF]
- [SIP-REP]
- [SIP-CT]
- [SDP-SPEC]

**Node Requirements:**
See [General Node Requirements](#).

**Discussion:** Oftentimes a UAC may place a call on hold, create a call with a second UAS, and then desire to transfer the call on hold with the first UAS to the second UAS. This can be accomplished by placing the second UAS on hold by sending an INVITE request indicating one-way audio, and then sending a REFER request to the first UAS. This REFER request will contain the SIP-URI of the second UAS in its Refer-To header. The first UAS will then send an INVITE request containing a Replaces header to the second UAS and thus replace the existing session between the UAC and the second UAS. Once the session has been established, the two UAS’s will end their existing sessions with the UAC by sending BYE requests.

**Test Setup:**
[Common Test Setup 1](#)

**Procedure:**

**Part A:**
1. UA2 calls UA1.
2. UA1 answers the call from UA2.
3. UA1 places the call with UA2 on hold.
4. UA1 calls UA3.
5. UA3 answers the call from UA1.
6. UA1 places the call with UA3 on hold.
7. UA1 transfers the call with UA2 to UA3.
8. Put UA1 back on hook.
9. Observe traffic on all networks.

**Observable Results:**

- **Part A**
  - **Step 2:** UA1 must answer the call from UA2. There must be audio flowing in both directions between UA1 and UA2.
  - **Step 3:** UA1 must place the call with UA2 on hold. There must not be audio flowing in either direction between UA1 and UA2.
  - **Step 4:** UA1 must call UA3.
Step 5: There must be audio flowing in both directions between UA1 and UA3.
Step 6: UA1 must place the call with UA3 on hold. There must not be audio flowing in either direction between UA1 and UA3.
Step 7: UA1 must transfer the call with UA2 to UA3. There must be audio flowing in both directions between UA2 and UA3. There must not be audio flowing in either direction between UA1 and UA2. There must not be audio flowing in either direction between UA1 and UA3.

Possible Problems:

- None
Test SIPFEAT.1.6: Call Forwarding Unconditional

Purpose: This test verifies that a SIP UA can successfully establish a call that is unconditionally forwarded.

References:

- [SIP-SPEC]

Node Requirements:
See General Node Requirements.

Discussion: Often times a user may want all calls destined for him to be redirected to another address. This can be accomplished by configuring the user’s proxy server to redirect all calls destined for him. After this has been configured, when another user sends an INVITE request to the proxy requesting a call with the user, the proxy will first respond with a 181 Call is Being Forwarded response, and then the proxy will forward the INVITE to the address that the user wished his calls to be forwarded to. From there the call setup will proceed normally.

Test Setup:

Common Test Setup 1

Procedure:

Part A:
1. UA1 calls UA2.
2. PS1 forwards the call from UA1 to UA3.
3. UA3 accepts the call.
4. Put UA1 back on hook.
5. Observe traffic on all networks.

Observable Results:

- Part A
  - Step 1: UA1 must call UA2.
  - Step 2: PS1 must forward the call from UA1 to UA3.
  - Step 3: There must be audio flowing in both directions between UA1 and UA3.
  - Step 5: There must not be audio flowing in either direction between UA1 and UA3.

Possible Problems:

- None
Test SIPFEAT.1.7: Call Forwarding - Busy

**Purpose:** This test verifies that a SIP UA can successfully forward an incoming call that is received while in a call with another UA.

**References:**
- [SIP-SPEC]

**Node Requirements:**
See [General Node Requirements](#).

**Discussion:** A user may desire to have calls forwarded to another address if he is busy in another call at the time. This can be accomplished by configuring his proxy server to forward the call in the event that his phone is busy. Thus, when some other user sends an INVITE request to the proxy server destined for the user, if the user’s phone sends back a 486 Busy Here response, the proxy will first send back a 181 Call is Being Forwarded response to the caller, and then forward the INVITE on to the other contact as configured in the server. From this point on the call setup can proceed as normal.

**Test Setup:**
- [Common Test Setup 1](#)

**Procedure:**

**Part A:**
1. UA1 calls UA2.
2. PS1 forwards the call from UA1 to UA2.
3. UA2 rejects the call from UA1 with a “busy” response.
4. PS1 forwards the call from UA1 to UA3.
5. UA3 answers the call from UA1.
6. Put UA1 back on hook.
7. Observe traffic on all networks.

**Observable Results:**
- **Part A**
  - **Step 1:** UA1 must call UA2.
  - **Step 2:** PS1 must forward the call from UA1 to UA2.
  - **Step 4:** PS1 must forward the call from UA1 to UA3.
  - **Step 5:** There must be audio traffic flowing in both directions between UA1 and UA3.
  - **Step 7:** There must not be audio traffic flowing in either direction between UA1 and UA3.

**Possible Problems:**
- None.
Test SIPFEAT.1.8: Call Forwarding – No Answer

Purpose: This test verifies that a SIP UA can successfully establish a call that is forwarded due to an unresponsive UAS.

References:
- [SIP-SPEC]

Node Requirements:
See General Node Requirements.

Discussion: A user may desire to have calls forwarded to another address if he does not answer the call in some time period. This can be accomplished by configuring his proxy server to forward the call in the event that some timeout occurs. Thus, when some other user sends an INVITE request to the proxy server destined for the user, if the user’s phone sends back a 180 Ringing response but never sends the 200 OK confirmation response, after the timeout period has elapsed the proxy will send a CANCEL request to the user’s phone. Once the original call has been successfully canceled, the proxy will send a 181 Call is Being Forwarded to the caller, and then forward the INVITE to the other contact as configured in the proxy. From there the call setup will proceed as normal.

Test Setup:
Common Test Setup 1

Procedure:

Part A:
1. UA1 calls UA2.
2. PS1 forwards the call from UA1 to UA2.
3. Allow a timeout to occur.
4. PS1 forwards the call from UA1 to UA3.
5. UA3 answers the call from UA1.
6. Put UA1 back on hook.
7. Observe traffic on all networks.

Observable Results:

- Part A
  - Step 1: UA1 must call UA2.
  - Step 2: PS1 must forward the call from UA1 to UA2.
  - Step 4: PS1 must forward the call from UA1 to UA3.
  - Step 5: There must be audio flowing in both directions between UA1 and UA3.
  - Step 7: There must not be audio flowing in either direction between UA1 and UA3.

Possible Problems:
- None.
Test SIPFEAT.1.9: 3-Way Conference – Third Party is Added

Purpose: This test verifies that a SIP UA can successfully add a third party to an ongoing call, thus creating a conference call.

References:

- [SIP-SPEC]
- [SIP-UAC]

Node Requirements:
See General Node Requirements.

Discussion: It is common to want to add a third party to an existing call, thus creating a 3-way conference call. One way to achieve this is if one of the users in the on-going call is capable of local mixing. If this is the case, the user can first send a re-INVITE the other user in the on-going call which changes the contact address as well as including the “isfocus” feature tag to indicate that it will be performing the mixing during the conference. Once the re-INVITE is accepted, the user can then send an INVITE request to the third party that will join the on-going call. This INVITE request will contain the same Contact address and “isfocus” feature tag contained in the previously sent re-INVITE.

Test Setup:
Common Test Setup 1

Procedure:

Part A:
1. UA2 calls UA1.
2. UA1 answers the call.
3. UA1 add UA3 to the call with UA2.
4. Observe traffic on all networks.

Observable Results:

- Part A
  Step 1: UA2 must call UA1.
  Step 2: There must be audio flowing in both directions between UA1 and UA2.
  Step 3: UA1 must add UA3 to the call with UA2.
  Step 4: There must be audio flowing in both directions between UA1 and UA2. There must be audio flowing in both directions between UA1 and UA3. Audio originating from UA2 must be heard by UA3. Audio originating from UA3 must be heard by UA2.

Possible Problems:

- UA1 may also put UA2 on hold before calling UA3, or may first call UA3 before sending the re-INVITE to U
Test SIPFEAT.1.10: Find-Me

Purpose: This test verifies that a SIP UA can successfully establish a call with a UAS that is behind a forking proxy that makes several attempts to contact the other party using a list of contact addresses.

References:

- [SIP-SPEC]

Node Requirements:
See General Node Requirements.

Discussion: A user with multiple contacts registered to a single AOR may wish all of these contacts to be tried when a user attempts to call them even if there is no response from the first contact tried. This can be accomplished by using a proxy that supports forking. The proxy must be configured in such a way that it has a list of contacts to attempt to contact when a call for a certain AOR is received. Once a call is received, the proxy will first forward the request to the first contact, and if a timeout occurs or it receives a non-200 final response, it will proceed to the next contact. Upon receiving the first final 200 response, the call can be established and the proxy will not proceed any further in the contact list.

Test Setup: Common Test Setup 1

Procedure:

Part A:
1. Ensure that UA3 and UA4 are registered to UA2’s AOR.
2. UA1 calls UA2.
3. PS1 forwards the call from UA1 to UA2.
4. Allow a timeout to occur.
5. PS1 forwards the call from UA1 to UA3.
6. UA3 rejects the call from UA1 with a “busy” response.
7. PS1 forwards the call from UA1 to UA4.
8. UA4 accepts the call from UA1.
9. Put UA1 back on hook.
10. Observe traffic on all networks.

Observable Results:

- Part A
  
  Step 2: UA1 must call UA2.
  Step 3: PS1 must forward the call from UA1 to UA2.
  Step 5: PS1 must forward the call from UA1 to UA3.
  Step 7: PS1 must forward the call from UA1 to UA4.
  Step 8: There must be audio flowing in both directions between UA1 and UA4.
  Step 10: There must not be audio flowing in either direction between UA1 and UA4.

Possible Problems:
• The proxy may also be configured to do parallel forking.
Test SIPFEAT.1.11: Call Management (Incoming Call Screening)

**Purpose:** This test verifies that a SIP UA can successfully authenticate itself to a proxy when requested to do so by a called party, and then contact an announcement server to listen to a recording after the call is rejected.

**References:**
- [SIP-SPEC]
- [SIP-DIAL]

**Node Requirements:**
See General Node Requirements.

**Discussion:** A user that wishes to have more control over which calls get forwarded to his phone can configure his proxy to perform incoming call screening. A proxy configured in this way has a list of callers that the user will accept calls from, and will reject all others. Incoming INVITE requests must be authenticated before being screened. If a call is sent directly to the user, his phone should respond with a 305 Use Proxy response containing the proxy’s SIP-URI in the Contact header. INVITE requests that do not meet the screening criteria will be responded to by the proxy with a 403 Screening Failure (Terminating) response that may include an Error-Info header. This header may contain the address of an announcement server that the rejected user can call in order to hear an announcement explaining why the call was rejected.

**Test Setup:**
[Common Test Setup 1]

**Procedure:**

**Part A:**
1. Ensure that UA1 is not in UA2’s accepted incoming caller list.
2. UA1 calls UA2.
3. UA2’s incoming call screening feature rejects the call from UA1.

**Observable Results:**

- **Part A**
  
  **Step 2:** UA1 must call UA2.
  
  **Step 4:** UA2’s incoming call screening feature must reject the call from UA1.

**Possible Problems:**

- None.
Test SIPFEAT.1.12: Call Management (Outgoing Call Screening)

Purpose: This test verifies that a SIP UA has the ability to establish and terminate a session when sending or receiving compact header field names directly with another UA.

References:
- [SIP-SPEC]

Node Requirements:
See General Node Requirements.

Discussion: Administrators of a network may wish to screen on-going calls made by users in their network. This can be accomplished by configuring the proxy server of the network to perform outgoing call screening. A proxy configured in this way has a list of contact that each user is allowed to call, and will reject all others. Users wishing to make a call through the proxy must first authenticate before attempting a call. Calls that to not meet the proper criteria will be responded to with a 403 Screening Failure (Originating). This response may include a SIP-URI in an Error-Info header that the user may contact to listen to an announcement.

Test Setup:
Common Test Setup 1

Procedure:

Part A:
1. Ensure that UA2 is not in UA1’s accepted outgoing caller list.
2. UA1 calls UA2.
3. UA1’s outgoing call screening feature rejects the call to UA2.

Observable Results:

- Part A
  Step 2: UA1 must call UA2.
  Step 4: UA1’s outgoing call screening feature must reject the call from UA1 to UA2.

Possible Problems:

- None.
Test SIPFEAT.1.13: Call Parking

Purpose: This test verifies that a SIP UA has the ability to park an established call on a call park server.

References:
- [SIP-SPEC]
- [SIP-REF]
- [SIP-REP]

Node Requirements:
See General Node Requirements.

Discussion: Call parking allows a user to transfer an ongoing call to a call park server where the other party will wait until a third user picks the call up. While parked, the user on hold may hear music from the call park server. To achieve this setup, the user wishing to park the call must include a Refer-To header containing the other user’s SIP-URI and the existing session’s identifiers in a REFER request sent to the call park server. Following this the call park server will send an INVITE request with a Replaces, replacing the session between the two users. Once the call is established, the user will hear on-hold music played by the call park server.

Test Setup:

Common Test Setup 1

Procedure:

Part A:
1. UA1 calls UA2.
2. UA2 accepts the call from UA1.
3. UA1 parks the call with UA2 on CPS1.
4. Put UA1 back on hook.
5. Observe traffic on all networks.

Observable Results:

- Part A
  - Step 1: UA1 must call UA2.
  - Step 2: There must be audio flowing in both directions between UA1 and UA2.
  - Step 3: UA1 must park the call with UA2 on CPS1.
  - Step 5: There must not be audio flowing in either direction between UA1 and UA2.

Possible Problems:

- None.
Test SIPFEAT.1.14: Parked Call Pickup

Purpose: This test verifies that a SIP UA has the ability to retrieve a call parked on a call park server.

References:
- [SIP-SPEC]
- [SIP-REF]
- [SIP-REP]

Node Requirements:
See General Node Requirements.

Discussion: After a call has been parked on a call park server, another user may wish to pickup up the call. This can be accomplished by sending an INVITE request to the parked user with a Replaces header containing the dialogue identifiers of the session between the parked user and the call park server. Once this INVITE is accepted, the then parked user will send a BYE request, ending its session with the call park server.

Test Setup:
- Common Test Setup 1

Procedure:

Part A:
1. UA2 calls UA3.
2. UA3 accepts the call from UA2.
3. UA2 parks the call with UA3 on CPS1.
4. UA1 picks up the call with UA3 on CPS1.
5. Observe traffic on all networks.

Observable Results:

- Part A
  - Step 1: UA2 must call UA3
  - Step 2: There must be audio flowing in both directions between UA2 and UA3.
  - Step 3: There must not be audio flowing in either direction between UA2 and UA3.
  - Step 4: UA1 must pick up the call with UA3 on CPS1.
  - Step 5: There must be audio flowing in both directions between UA1 and UA3.

Possible Problems:

- None.
Test SIPFEAT.1.15: Automatic Redial

**Purpose:** This test verifies that a SIP UA can successfully subscribe to the call state of a UAS so as to call them when they become available.

**References:**

- [SIP-SPEC]
- [SIP-DIAL]

**Node Requirements:**

See [General Node Requirements](#).

**Discussion:** When a user call’s another user and finds that user to be unavailable, it is common to want to call them as soon as they are able to receive calls again. This can be accomplished by sending a SUBSCRIBE request using the dialogue package to the callee. Once the subscription is accepted, the callee will send NOTIFY requests to the user upon changes to its call state. When the user receives notification that the callee is available, an INVITE request can be sent and the call can be established.

**Test Setup:**

[Common Test Setup 1](#)

**Procedure:**

**Part A:**

1. Set UA2 to a busy state.
2. UA1 calls UA2.
3. UA2 rejects the call from UA1 with a “busy” response.
4. UA2 becomes available to call.
5. UA1’s automatic redial feature activates and calls UA2 from UA1.
6. UA2 accepts the call from UA1.
7. Observe traffic on all networks.

**Observable Results:**

- **Part A**
  
  **Step 2:** UA1 must call UA2.
  **Step 3:** UA2 must reject the call from UA1 with a “busy” response.
  **Step 5:** UA1’s automatic redial feature must activate and call UA2 from UA1.
  **Step 7:** There must be audio flowing in both directions between UA1 and UA2.

**Possible Problems:**

- None.
Test SIPFEAT.1.16: Click to Dial

**Purpose:** This test verifies that a SIP UA can successfully receive a REFER request from a web client and attempt a call as directed by the request.

**References:**
- [SIP-SPEC]
- [SIP-REF]

**Node Requirements:**
See [General Node Requirements](#).

**Discussion:** A useful feature is the ability to click a SIP-URI in a browser and have the SIP client running on the computer make the call. This can be accomplished by having the browser send a REFER request to the SIP client with a Refer-To header containing the SIP-URI in the link. Once the SIP client receives the REFER request, it will send an INVITE request to the SIP-URI.

**Test Setup:**
- [Common Test Setup 1](#)

**Procedure:**

**Part A:**
1. The owner of UA1 activates its click to dial feature in a web browser on host other than UA1.
2. UA1’s SIP phone calls UA2.
3. UA2 accepts the call from UA1’s SIP phone.
4. Observe traffic on all networks.

**Observable Results:**

- **Part A**
  - **Step 1:** The owner of UA1 must activate its click to dial feature in a web browser on host other than UA1.
  - **Step 2:** UA1’s SIP phone must call UA2.
  - **Step 5:** There must be audio flowing in both direction between UA1 and UA2.

**Possible Problems:**

- None.