Vendor X
Company Y
123 North St
Anywhere, NH 12345

Mr./Mrs. Vendor X,

Enclosed are the results from the Session Initiation Protocol (SIP) Features testing performed on:

Device tested: Device Z
SW/FW: 1.2.3

The tests performed are part of the SIP Features Test Suite, which can be found at:

ftp://public.iol.unh.edu/pub/voip/testsuites/sip_features_test_suite_0.3.pdf

During the testing process, the following issues were uncovered:

<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test SIPFEAT_1.1</td>
<td>The DUT failed to recognize the connection address “0.0.0.0” in the SDP offer of the INVITE from UA1 as a request to place the call on-hold.</td>
</tr>
<tr>
<td>Test SIPFEAT_1.5</td>
<td>The DUT failed to recognize the “Replaces” header field in the INVITE request from UA2.</td>
</tr>
<tr>
<td>Test SIPFEAT_1.13</td>
<td>The DUT failed to decode the DTMF tones from UA1 indicating a request to park the call with UA2 at CPS1.</td>
</tr>
</tbody>
</table>

The DUT was tested against the following devices during the testing process:

<table>
<thead>
<tr>
<th>Common Test Setup 1</th>
<th>Model</th>
<th>IP Address</th>
<th>FQDN</th>
</tr>
</thead>
<tbody>
<tr>
<td>UA1</td>
<td>Device 1</td>
<td>123.456.789.1</td>
<td>device1.iol.unh.edu</td>
</tr>
<tr>
<td>UA2</td>
<td>Device 2</td>
<td>123.456.789.2</td>
<td>device2.iol.unh.edu</td>
</tr>
<tr>
<td>UA3</td>
<td>Device 3</td>
<td>123.456.789.3</td>
<td>device3.iol.unh.edu</td>
</tr>
<tr>
<td>UA4</td>
<td>Device 4</td>
<td>123.456.789.4</td>
<td>device4.iol.unh.edu</td>
</tr>
<tr>
<td>PS1</td>
<td>Device 5</td>
<td>123.456.789.5</td>
<td>device5.iol.unh.edu</td>
</tr>
<tr>
<td>B2BUA</td>
<td>Device 6</td>
<td>123.456.789.6</td>
<td>device6.iol.unh.edu</td>
</tr>
<tr>
<td>MOHS</td>
<td>Device 7</td>
<td>123.456.789.7</td>
<td>device7.iol.unh.edu</td>
</tr>
<tr>
<td>CPS1</td>
<td>Device 8</td>
<td>123.456.789.8</td>
<td>device8.iol.unh.edu</td>
</tr>
<tr>
<td>AS1</td>
<td>Device 9</td>
<td>123.456.789.9</td>
<td>device9.iol.unh.edu</td>
</tr>
</tbody>
</table>
Note: All software versions listed are official releases and/or releases that are authorized by a representative of the device, on behalf of the company, for use in this testing. Results documented in this report are not guaranteed or assumed on other software versions or hardware.

The following logical topology was used during testing:

As always, we welcome any comments regarding this Test Suite. If you have any questions about the test procedures or results, please feel free to contact me via e-mail at tester@iol.unh.edu or by phone at +1-603-862-0186.

Regards,
Tester

Reviewed by,
Reviewer

Digitally signed by
UNH-IOL
Date:
2009.07.2
2 14:37:30
-04'00'
Digital Signature Information

This document was created using an Adobe digital signature. A digital signature helps to ensure the authenticity of the document, but only in this digital format. For information on how to verify this document’s integrity proceed to the following site:

http://www.iol.unh.edu/certifyDoc/

If the document status still indicates “Validity of author NOT confirmed”, then please contact the UNH-IOL to confirm the document’s authenticity. To further validate the certificate integrity, Adobe 6.0 should report the following fingerprint information:

MD5 Fingerprint: F6E2 1B99 28AD 0D25 E77E ADE5 479A 1E05
SHA-1 Fingerprint: AD30 8B08 DD3B B2E3 9362 46E9 3427 BE47 1D49 890B

The following table contains possible results and their meanings. If a test passes, the DUT passes with all test units involved. If the test fails, the report will indicate the test unit(s) that the failure involved.

<table>
<thead>
<tr>
<th>Result</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>PASS</td>
<td>No Interoperability problems were discovered with any Test Devices.</td>
</tr>
<tr>
<td>FAIL</td>
<td>Interoperability problems were encountered with certain Test Devices. This resulted in undesirable behavior.</td>
</tr>
<tr>
<td>WARN</td>
<td>Interoperability problems were encountered with certain Test Devices. This resulted in behavior that is not recommended.</td>
</tr>
<tr>
<td>N/A</td>
<td>Not Applicable. This test is not applicable for the DUT.</td>
</tr>
<tr>
<td>N/S</td>
<td>Not Supported. This test was not run due to features not implemented on the DUT.</td>
</tr>
<tr>
<td>N/T</td>
<td>Not tested. This test was not run.</td>
</tr>
</tbody>
</table>
### Test 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 hook.
6. Observe traffic on all networks.

### Test Results

A. A call was established between UA2 and UA1. UA1 attempted to place the call with UA2 on-hold by transmitting an INVITE request containing an SDP offer with a connection address of “0.0.0.0” to PS1. PS1 did not recognize the connection address of “0.0.0.0” as a request to place the session on-hold and rejected the request with a 488 Not Acceptable Here message. Therefore, the call was not placed on-hold.
### Test # | Result
--- | ---
SIPFEAT.1.2 | Consultation Hold | A | PASS

**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.

**Comments on Test 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 hook.
9. Observe traffic on all networks.

**Comments on Test Results**

A. There was audio flowing in both directions between UA1 and UA2. UA1 placed the call with UA2 on-hold. There was no audio flowing in either direction between UA1 and UA2. There was audio flowing in both directions between UA1 and UA3. There was no audio flowing in either direction between UA1 and UA3. UA1 took the call with UA2 off-hold. There was audio flowing in both directions. There was no audio flowing in either direction between UA1 and UA2.
### Test 

<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.3</td>
<td>Music On Hold</td>
</tr>
</tbody>
</table>

**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.

**Comments on Test 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 using MOHS1 to play music on hold audio to UA2.
5. UA1 takes the call with UA2 off hold.
6. Put UA2 back on hook.
7. Observe traffic on all networks.

**Comments on Test Results**

A. There was audio flowing in both directions between UA1 and UA2. UA1 placed the call with UA2 on hold using MOHS1 as its music-on-hold server. There was no audio flowing in either direction between UA1 and UA2. There was audio flowing from MOHS1 to UA2. UA1 took the call with UA2 off-hold. There was audio flowing in both directions between UA1 and UA2. There was no audio flowing from MOHS1 to UA2. There was no audio flowing in either direction between UA1 and UA2.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.4</td>
<td>Transfer - Unattended</td>
</tr>
<tr>
<td>A</td>
<td>PASS</td>
</tr>
</tbody>
</table>

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

**Comments on Test 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. UA3 answers the call.
6. Observe traffic on all networks.

**Comments on Test Results**

A. UA1 called UA2. UA1 transferred the call with UA2 to UA3.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.5</td>
<td>Transfer - Attended</td>
</tr>
</tbody>
</table>

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

**Comments on Test 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 UA3 to UA2.
8. Put UA1 back on-hook.
9. Observe traffic on all networks.

**Comments on Test Results**

A. UA1 answered the call from UA2. There was audio flowing in both directions between UA1 and UA2. UA1 placed the call with UA2 on-hold. UA1 called UA3. UA1 placed the call with UA3 on-hold. UA1 transmitted a REFER request to UA2 with a “Refer-To” header field containing UA3’s URI. UA2 transmitted an INVITE request to PS1 with a “Replaces” header field containing the ID for the previous session between UA1 and UA3. PS1 did not recognize the “Replaces” header field and therefore did not forward the INVITE request to UA3.
## Test #

<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.6</td>
<td>A</td>
<td>This test verifies that a SIP UA can successfully establish a call that is unconditionally forwarded.</td>
</tr>
</tbody>
</table>

### Comments on Test Procedure

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

### Comments on Test Results

A. UA2 called UA1. PS1 forwarded the call from UA1 to UA3. There was audio flowing in both directions between UA1 and UA3. There was no audio flowing in either direction between UA1 and UA3.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.7</td>
<td></td>
<td>This test verifies that a SIP UA can successfully forward an incoming call that is received while in a call with another UA.</td>
</tr>
</tbody>
</table>

**Comments on Test 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.

**Comments on Test Results**

A. UA1 called UA2. PS1 forwarded the call from UA1 to UA3. PS1 forwarded the call from UA1 to UA3. There was audio traffic flowing in both directions between UA1 and UA3. There was no audio flowing in either direction between UA1 and UA3.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.8</td>
<td>PASS</td>
<td>This test verifies that a SIP UA can successfully establish a call that is forwarded due to an unresponsive UAS.</td>
</tr>
</tbody>
</table>

**Comments on Test 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 UA2 back on hook.
7. Observe traffic on all networks.

**Comments on Test Results**

A. UA1 called UA2. PS1 forwarded the call from UA1 to UA2. PS1 forwarded the call from UA1 to UA3. A call was established between UA1 and UA3.
### Test # | Result
---|---
SIPFEAT.1.9 | A | PASS

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

**Comments on Test Procedure**

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

**Comments on Test Results**

A. UA2 called UA1. There was audio flowing in both directions between UA1 and UA2. UA1 added UA3 to the call with UA2. There was audio flowing in both directions between UA1 and UA2. There was audio flowing in both directions between UA1 and UA2. Audio originated from UA2 must be heard by UA3. Audio originated from UA3 must be heard by UA2.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.10</td>
<td>Find-Me</td>
<td>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.</td>
</tr>
</tbody>
</table>

Comments on Test 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.

Comments on Test Results

**A.** UA1 called UA2. PS1 forwarded the call from UA1 to UA2. PS1 forwarded the call from UA1 to UA3. PS1 forwarded the call from UA1 to UA4. There must be audio flowing in both directions between UA1 and UA4. There was no audio flowing in either direction between UA1 and UA4.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.11</td>
<td>A</td>
</tr>
</tbody>
</table>

**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.

**Comments on Test 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.

**Comments on Test Results**

A. UA1 called UA2. UA2’s incoming call-screening rejected the call from UA1.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.12</td>
<td>Call Management (Outgoing Call Screening)</td>
</tr>
</tbody>
</table>

**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.

**Comments on Test 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.

**Comments on Test Results**

A. UA1 called UA2. UA1’s outgoing call-screening feature rejected the call from UA1 to UA2.
<table>
<thead>
<tr>
<th>Test #</th>
<th>Result</th>
<th>Purpose</th>
<th>Comments on Test Procedure</th>
<th>Comments on Test Results</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.13</td>
<td></td>
<td>This test verifies that a SIP UA has the ability to park an established call on a call park server.</td>
<td>Part A:&lt;br&gt;1. UA2 calls UA1. 2. UA1 accepts the call from UA2. 3. UA1 parks the call with UA2 at CPS1. 4. Put UA1 back on-hook. 5. Observe traffic on all networks.</td>
<td>A. UA2 called UA1. There was audio flowing in both directions between UA1 and UA2. UA1 attempts to park the call with UA2 at CPS1 by transmitting DTMF digits as tones through the media channel. PS1 was unable to decode the DTMF digits and therefore did not transmit an INVITE request to UA2 to replace the session with UA1.</td>
</tr>
</tbody>
</table>

**Purpose**

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

**Comments on Test Procedure**

**Part A:**

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

**Comments on Test Results**

A. UA2 called UA1. There was audio flowing in both directions between UA1 and UA2. UA1 attempts to park the call with UA2 at CPS1 by transmitting DTMF digits as tones through the media channel. PS1 was unable to decode the DTMF digits and therefore did not transmit an INVITE request to UA2 to replace the session with UA1.
### Test #

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.14</td>
<td>A PASS</td>
</tr>
</tbody>
</table>

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

**Comments on Test 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.

**Comments on Test Results**

A. UA2 called UA3. There was audio flowing in both directions between UA2 and UA3. There was no audio flowing in either direction between UA2 and UA3. UA1 picked up the call with UA3 on CPS1. There was audio flowing in both directions between UA1 and UA3.
### Test #

<table>
<thead>
<tr>
<th>Test</th>
<th>Result</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIPFEAT.1.15</td>
<td>A</td>
</tr>
</tbody>
</table>

#### 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.

#### Comments on Test Procedure

**Part A:**
1. Set UA2 to a busy state.
2. UA1 calls UA2.
3. UA2 rejects the call 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.

#### Comments on Test Results

A. UA1 called UA2. UA2 rejected the call from UA1 with a “busy” response. UA1’s automatic redial feature activated and called UA2 from UA1. There was audio flowing in both directions between UA1 and UA2.
### Test Procedure

**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.

#### Comments on Test 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.

#### Comments on Test Results

A. The owner of UA1 computer activated its click-to-dial feature in a web browser on host other than UA1. UA1’s SIP phone called UA2. There was audio flowing in both directions between UA1 and UA2.