Voice over IP Consortium
SIP Torture Test Suite

Version 1.5.2

Last Updated: October 13, 2006

VoIP Consortium
University of New Hampshire
Research Computing Center
InterOperability Laboratory

121 Technology Drive, Suite 2
Durham, NH 03824
Phone: +1-603-862-0186
Fax: +1-603-862-4181

www.iol.unh.edu

© 2006 University of New Hampshire InterOperability Laboratory
# Modification Report

<table>
<thead>
<tr>
<th>Version</th>
<th>Date</th>
<th>Editor(s)</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.0</td>
<td>June 27, 2006</td>
<td>Niels Widger</td>
<td>Initial draft created</td>
</tr>
<tr>
<td>1.1</td>
<td>July 13, 2006</td>
<td>Allen Latham</td>
<td>Section 1 Added</td>
</tr>
<tr>
<td>1.2</td>
<td>July 19, 2006</td>
<td>Allen Latham</td>
<td>Sections 2-5 added, formatting revised</td>
</tr>
<tr>
<td>1.3</td>
<td>July 21, 2006</td>
<td>Allen Latham</td>
<td>Formatting Revised</td>
</tr>
<tr>
<td>1.4</td>
<td>August 7, 2006</td>
<td>Allen Latham</td>
<td>Content Revised</td>
</tr>
<tr>
<td>1.5</td>
<td>August 15, 2006</td>
<td>Allen Latham</td>
<td>Content, Grammar, and Formatting Revised</td>
</tr>
<tr>
<td>1.5.1</td>
<td>October 10, 2006</td>
<td>Allen Latham</td>
<td>General Review</td>
</tr>
<tr>
<td>1.5.2</td>
<td>October 13, 2006</td>
<td>Allen Latham, Niels Widger, Bryan Pellegrino</td>
<td>General Review, Revised for Readability</td>
</tr>
</tbody>
</table>
Acknowledgments

The University of New Hampshire 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>Allen Latham</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>James Swan</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>Lincoln Lavoie</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>Bryan Pellegrino</td>
<td>University of New Hampshire</td>
</tr>
<tr>
<td>Niels Widger</td>
<td>University of New Hampshire</td>
</tr>
</tbody>
</table>
Table of Contents

Modification Report.......................................................................................................................... i
Acknowledgments.............................................................................................................................. ii
Table of Contents .............................................................................................................................. iii
Introduction........................................................................................................................................ vi
References .......................................................................................................................................... viii
Terms: Definitions and Abbreviations .............................................................................................. ix
Definitions ........................................................................................................................................... ix
Abbreviations ..................................................................................................................................... x
Test Setups .......................................................................................................................................... xi
Test Setup 1: General Test Setup ....................................................................................................... xi

Group 1: Valid Messages .................................................................................................................... 1
Test # 1.1: Accepting a Torturous INVITE ....................................................................................... 2
Test # 1.2: Accepting Valid Characters ............................................................................................ 3
Test # 1.3: Valid use of “%” as an Escape ....................................................................................... 4
Test # 1.4: Escaped Nulls in URIs .................................................................................................. 5
Test # 1.4: Escaped Nulls in URIs .................................................................................................. 5
Test # 1.5: Valid Use of “%” When Not as an Escape ..................................................................... 6
Test # 1.6: Message with No Linear White Space in the Display Name ........................................... 7
Test # 1.7: Long values in Header Fields......................................................................................... 8
Test # 1.8: Extra Trailing Octets .................................................................................................... 9
Test # 1.9: Semicolon Separated Parameters in the URI Userpart.................................................... 10
Test # 1.10: Unknown Transport Types ......................................................................................... 11
Test # 1.11: Multipart MIME Messages ......................................................................................... 12
Test # 1.12: Unusual Reason Phrase .............................................................................................. 13
Test # 1.13: No Reason Phrase ...................................................................................................... 14

Group 2: Invalid Messages .............................................................................................................. 15
Test # 2.1: Extraneous Header Field Separators ........................................................................... 16
Test # 2.2: Content Length Larger than the Message ..................................................................... 17
Test # 2.3: Negative Content Length ............................................................................................ 18
Test # 2.4: Request Scalar Fields with Overlarge Values ............................................................... 19
Test # 2.5: Response Scalar Fields with Overlarge Values .......................................................... 20
Test # 2.6: Unterminated Quoted String ......................................................................................... 21
Test # 2.7: “<>” Enclosing the URI .............................................................................................. 22
Test # 2.8: Malformed SIP Request URI ....................................................................................... 23
Test # 2.9: Multiple SP Separating Request Line Elements .......................................................... 24
Test # 2.10: SP Characters at the End of the Request Line ............................................................ 25
Test # 2.11: Escaped Headers (%) in the Request URI ................................................................. 26
Test # 2.12: Invalid Time Zone .................................................................................................... 27
Test # 2.13: Name-Addr Format URI without “<>” ..................................................................... 28
Test # 2.14: Spaces Within Addr-Spec ....................................................................................... 29
Test # 2.15: Non-token Characters in the Display Name ............................................................... 30
Test # 2.16: Unknown Protocol Version ....................................................................................... 31
Test # 2.17: Request Line and CSeq Method Mismatch ................................................................. 32
Test # 2.18: Unknown Method with CSeq Method Mismatch ...................................................... 33
Test # 2.19: Overlarge Response Code ......................................................................................... 34

Group 3: Transaction Layer Semantics .......................................................................................... 35

VoIP Consortium ................................. iii ................................. SIP Torture Test Suite
The University of New Hampshire
InterOperability Laboratory

Test # 3.1: Missing Transaction Identifiers................................................................. 36

Group 4: Application Layer Semantics.................................................................. 37
Test # 4.1: Missing Required Header Fields................................................................. 38
Test # 4.2: Request URI with Unknown Scheme ......................................................... 39
Test # 4.3: Request URI with Known But Atypical Scheme ...................................... 40
Test # 4.4: Unknown URI in Header Scheme............................................................... 41
Test # 4.5: Unsupported Requires and Proxy-Requires .............................................. 42
Test # 4.6: Unknown Content Type ........................................................................... 43
Test # 4.7: Unknown Authorization Scheme .............................................................. 44
Test # 4.8: Multiple Values in Single Value Fields .................................................... 45
Test # 4.9: Multiple Content Length Values .............................................................. 46
Test # 4.10: Response with Broadcast Via Header Field Value .................................. 47
Test # 4.11: Max-Forward of Zero.................................................................................... 48
Test # 4.12: REGISTER with a Contact Header Parameter ........................................ 49
Test # 4.13: REGISTER with a URL Parameter .......................................................... 50
Test # 4.14: REGISTER with a URL Escaped Header ................................................. 51
Test # 4.15: Unacceptable Accept Offering ............................................................... 52

Group 5: Backwards Compatibility ...................................................................... 53
Test # 5.1: INVITE with RFC 2543 Syntax................................................................. 54

Appendix 1: Test Messages ...................................................................................... 55
Test Message 1.1......................................................................................................... 56
Test Message 1.2......................................................................................................... 57
Test Message 1.3......................................................................................................... 57
Test Message 1.4......................................................................................................... 58
Test Message 1.5......................................................................................................... 58
Test Message 1.6......................................................................................................... 58
Test Message 1.7......................................................................................................... 59
Test Message 1.8......................................................................................................... 60
Test Message 1.9......................................................................................................... 61
Test Message 1.10....................................................................................................... 61
Test Message 1.11....................................................................................................... 62
Test Message 1.12....................................................................................................... 63
Test Message 1.13....................................................................................................... 63
Test Message 2.1......................................................................................................... 63
Test Message 2.2......................................................................................................... 64
Test Message 2.3......................................................................................................... 64
Test Message 2.4......................................................................................................... 65
Test Message 2.5......................................................................................................... 65
Test Message 2.6......................................................................................................... 65
Test Message 2.7......................................................................................................... 66
Test Message 2.8......................................................................................................... 66
Test Message 2.9......................................................................................................... 67
Test Message 2.10....................................................................................................... 67
Test Message 2.11....................................................................................................... 68
Test Message 2.12....................................................................................................... 68
Test Message 2.13....................................................................................................... 68
Test Message 2.14....................................................................................................... 69
Test Message 2.15....................................................................................................... 69
Test Message 2.16....................................................................................................... 69
Test Message 2.17....................................................................................................... 70
Test Message 2.18....................................................................................................... 70
Test Message 2.19....................................................................................................... 70
Test Message 3.1......................................................................................................... 70
Test Message 4.1......................................................................................................... 71
<table>
<thead>
<tr>
<th>Test Message 4.2</th>
<th>71</th>
</tr>
</thead>
<tbody>
<tr>
<td>Test Message 4.3</td>
<td>71</td>
</tr>
<tr>
<td>Test Message 4.4</td>
<td>71</td>
</tr>
<tr>
<td>Test Message 4.5</td>
<td>72</td>
</tr>
<tr>
<td>Test Message 4.6</td>
<td>72</td>
</tr>
<tr>
<td>Test Message 4.7</td>
<td>72</td>
</tr>
<tr>
<td>Test Message 4.8</td>
<td>73</td>
</tr>
<tr>
<td>Test Message 4.9</td>
<td>73</td>
</tr>
<tr>
<td>Test Message 4.10</td>
<td>73</td>
</tr>
<tr>
<td>Test Message 4.11</td>
<td>74</td>
</tr>
<tr>
<td>Test Message 4.12</td>
<td>74</td>
</tr>
<tr>
<td>Test Message 4.13</td>
<td>74</td>
</tr>
<tr>
<td>Test Message 4.14</td>
<td>75</td>
</tr>
<tr>
<td>Test Message 4.15</td>
<td>75</td>
</tr>
<tr>
<td>Test Message 5.1</td>
<td>75</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 suite of tests has been developed to help implementers evaluate the conformance of their SIP Endpoint Implementations.

The success metrics of these tests are based on the standards referenced in this document. Successful completion of all tests contained in this suite does not guarantee that the tested device will operate with other devices. However, these tests provide a reasonable level of confidence that the Device Under Test will function well in most multi-vendor environments.

Organization of Tests:
Each test contains an identification section that describes the test and provides cross-reference information. The discussion section covers background information and specifies why the test is to be performed. Tests are grouped in order to reduce setup time in the lab environment. Each test contains the following information:

Test Number
The Test Number associated with each test follows a simple grouping structure. Listed first is the Test Group Number followed by the test's number within the group. This allows for the addition of future tests to the appropriate groups of the test suite without requiring the rennumbering of the subsequent tests.

Purpose
The purpose is a brief statement outlining what the test attempts to achieve. This also includes background information on why one needs to perform such a test to show that the device complies with the standard.

References
The references section lists standards and other documentation that might be helpful in understanding and evaluating the test and results.

Resource Requirements
The requirements section specifies the hardware, and test equipment that will be needed to perform the test. The items contained in this section are special test devices or other facilities, which may not be available on all devices.

Last Modification
This specifies the date of the last modification to this test.

Test Layout
The setup section describes the configuration of the test environment. Small changes in the configuration should be included in the test procedure.

Discussion
The discussion section is optional. It 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.

Procedure
The procedure section of the test description contains the step-by-step instructions for carrying out the test. It provides a cookbook approach to testing, and may be interspersed with observable results.

Test Metrics
The test metrics section lists the necessary parameters for success in a given test. When multiple values are possible for a specific event, this section provides a short discussion on how to interpret them. The tests are structured so that failure of one test metric will result in a failure for the entire test, or a request to refer to comments.
Possible Problems
This section contains a description of known issues with the test procedure, which may affect test results in certain situations.
References


## Terms: Definitions and Abbreviations

### Definitions

<table>
<thead>
<tr>
<th>Term</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Callee</td>
<td>The endpoint at which the call is received.</td>
</tr>
<tr>
<td>Caller</td>
<td>The endpoint at which the call process is started.</td>
</tr>
<tr>
<td>Dialog</td>
<td>A dialog is 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 dialog is identified by a call identifier, local tag, and a remote tag.</td>
</tr>
<tr>
<td>Message</td>
<td>Data sent between SIP elements as part of the protocol. SIP messages are either requests or responses.</td>
</tr>
<tr>
<td>Method</td>
<td>The method is the primary function that a request is meant to invoke on a server. The method is carried in the request message itself. Example methods are INVITE and BYE.</td>
</tr>
<tr>
<td>Request</td>
<td>A SIP message sent from a client to a server, for the purpose of invoking a particular operation.</td>
</tr>
<tr>
<td>Response</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>Session</td>
<td>From the SDP specification: &quot;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.&quot; (RFC 2327 [1]) (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 origin field.</td>
</tr>
<tr>
<td>SIP Transaction</td>
<td>A SIP transaction occurs between a client and a server and comprises all messages from the first request sent from the client to the server up to a final (non-1xx) response sent from the server to the client. If the request is INVITE and the final response is a non-2xx, the transaction also includes an ACK to the response. The ACK for a 2xx response to an INVITE request is a separate transaction.</td>
</tr>
<tr>
<td>UA (User Agent)</td>
<td>A logical entity that can act as both a user agent client and user agent server</td>
</tr>
<tr>
<td>UAC (User Agent Client)</td>
<td>A user agent client is a logical entity that creates a new request, and then uses the client transaction state machinery to send it. The role of UAC lasts only for the duration of that transaction. In other words, if a piece of software initiates a request, it acts as a UAC for the duration of that transaction. If it receives a request later, it assumes the role of a user agent server for the processing of that transaction.</td>
</tr>
<tr>
<td>UAS (User Agent Server)</td>
<td>A user agent server is a logical entity that generates a response to a SIP request. The response accepts, rejects, or redirects the request. This role lasts only for the duration of that transaction. In other words, if a piece of software responds to a request, it acts as a UAS for the duration of that transaction. If it generates a request later, it assumes the role of a user agent client for the processing of that transaction.</td>
</tr>
<tr>
<td>URI (Universal Resource Indicator)</td>
<td>Generally, a formatted string which provides information on how to access a particular resource. A SIP URI is in the form: sip:<a href="mailto:username@network-address.com">username@network-address.com</a> and is the primary method of locating users on the Internet.</td>
</tr>
</tbody>
</table>
### Abbreviations

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>LAN</td>
<td>Local Area Network</td>
</tr>
<tr>
<td>RTP</td>
<td>Real Time 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>UDP</td>
<td>User Datagram Protocol</td>
</tr>
<tr>
<td>URI</td>
<td>Uniform Resource Identifier</td>
</tr>
<tr>
<td>VLAN</td>
<td>Virtual LAN</td>
</tr>
<tr>
<td>WAN</td>
<td>Wide Area Network</td>
</tr>
</tbody>
</table>
Test Setups

Test Setup 1: General Test Setup

Unless specified within a particular test, all tests in this test suite use the same physical network configuration, network topology and device configuration. The DUT, within the context of this test suite, is an implementation of a SIP endpoint device conformant to RFC 3261.

Given the narrow scope of this test suite, there are only two primary components required for each test. The first is the Device Under Test (DUT); the second is a programatically configured SIP user agent (SIP Endpoint) capable of fine-tuning the contents of SIP elements such a headers, requests, transactions and dialogs.

Configuration Settings

<table>
<thead>
<tr>
<th>Parameter</th>
<th>SIP Endpoint</th>
<th>DUT</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP URI</td>
<td>sip:ep@sftf-endpoint</td>
<td>sip:dut@dut-endpoint</td>
</tr>
<tr>
<td>Proxy Server</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>Registrar</td>
<td>None</td>
<td>None</td>
</tr>
<tr>
<td>IP Address</td>
<td>10.10.0.85</td>
<td>10.10.0.11</td>
</tr>
<tr>
<td>Transport</td>
<td>UDP</td>
<td>UDP</td>
</tr>
<tr>
<td>Port</td>
<td>5060</td>
<td>5060</td>
</tr>
</tbody>
</table>
Group 1: Valid Messages

Scope:
The following tests focus on testing a device’s capability to accept messages that are designed to test the boundaries of what the device is willing to accept, while still being a valid message.

Overview:
This series of tests assess the DUT’s ability to identify and accept several atypical messages that include unusual characters, character usages, values, and reason phrases.
Test # 1.1: Accepting a Torturous INVITE

**Purpose:** This test assesses the DUT's ability to recognize and accept a valid SIP method with many unknown and unusual values.

**References:**

[1] Section 3.1.1.1 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 13, 2006

**Discussion:**
This invite is a valid and well-formed message. This test verifies that the DUT can accept defined SIP Methods with unknown values/headers while still responding in the appropriate manner (in this case, a “200 Ok”).

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send INVITE request from the SIP Endpoint to the DUT as defined by Test Message 1.1.

**Observable Results:**
1. Verify that the DUT responded with a “200 Ok”.

**Possible Problems:**
- None
Test # 1.2: Accepting Valid Characters

**Purpose:** This test assesses a DUT’s ability to identify and accept a wide range of valid characters.

**References:**
[1] Section 3.1.1.2 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 13, 2006

**Discussion:**
This test assesses the DUT’s ability to accept a large variety of valid characters. This includes the Method containing non-alpha characters, unusual characters in the URI and headers, and an unknown header field with non-alpha characters in its name and an unusual value.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send a request from the SIP Endpoint to the DUT as defined by Test Message 1.2.

**Observable Results:**
1. Verify the DUT responded with a “501 Not Implemented” listing the Methods that it understood in the “allow” header.

**Possible Problems:**
- Due to the “unusual URI” in the header line a proxy may generate a “404 Not Found” and not forward this request.
Test # 1.3: Valid use of “%” as an Escape

**Purpose:** This test verifies the DUT’s ability to accept and respond to Methods with “%” being used as an escape mechanism.

**References:**
[1] Section 3.1.1.3 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 13, 2006

**Discussion:**
This invite is a well-formed and valid message. The DUT must accept the % expansions as equivalent to the character encoded. Escapes should only occur in the URIs and other valid headings/ reason phrases.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send INVITE request from the SIP Endpoint to the DUT as defined by Test Message 1.3.

**Observable Results:**
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that the “%” escapes were properly parsed as their values.

**Possible Problems:**
- None
Test # 1.4: Escaped Nulls in URIs

Purpose: This test verifies that the DUT can identify and accept a Register with escaped nulls (%) in the URI user part.

References:
[1] Section 3.1.1.4 (RFC 4475)

Resource Requirements
• SIP Endpoint
• DUT

Last Modification: August 3, 2006

Discussion:
This message is well formed and should be accepted as such. The escaped nulls are only in the user part of the URI so no other components of this register should be affected by the escaping.

Test Layout:
• Test Setup 1

Procedure:
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 1.4.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that the DUT registered 2 distinct contacts and none of the usernames were shortened.

Possible Problems:
• A User-Agent might not properly responded to this request because it is a REGISTER normally intended for register servers/proxies, this is not a failure on the part of the User-Agent.
Test # 1.5: Valid Use of “%” When Not as an Escape

Purpose: This test verifies the DUT can identify and accept “%” when not used as an escape value.

References:
[1] Section 3.1.1.5 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion:
This is a well-formed message and should be accepted as such. It is important to note that the method is not equivalent to REGISTER and is unknown in this instance. This is the case with the rest of the header values as well.

Test Layout:
- Test Setup 1

Procedure:
1. Send the request from the SIP Endpoint to the DUT as defined by Test Message 1.5.

Observable Results:
1. Verify that the DUT either responded with a “501 Not Implemented” if acting as an endpoint or properly forwarded/ rejected the message depending on its definitions, if acting as a proxy.

Possible Problems:
- None
Test # 1.6: Message with No Linear White Space in the Display Name

Purpose: This tests the DUT’s ability to accept a URI with no linear white space between the Display Name and the “<”.

References:
[1] Section 3.1.1.6 (RFC 4475)

Resource Requirements:
• SIP Endpoint
• DUT

Last Modification: August 3, 2006

Discussion:
This is a well-formed request and should be accepted as such. This piece of grammar is incorrect by the 2002 version of RFC 3261, therefore failing this test does not technically violate the current standard. However, this message is considered valid and should be remedied in the revision of RFC 3261.

Test Layout:
• Test Setup 1

Procedure:
1. Send the OPTIONS request from the SIP Endpoint to the DUT defined by Test Message 1.6.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.

Possible Problems:
• None
Test # 1.7: Long values in Header Fields

Purpose: This tests the DUT’s ability to identify and accept requests with header fields that contain both many values and very long values.

References:
[1] Section 3.1.1.7 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion:
This is a well-formed request and should be accepted as such. While the via headings contain an excessive amount of values, this message is valid and should be accepted by the endpoint. It is important to note that the Call ID is one long token.

Test Layout:
- Test Setup 1

Procedure:
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 1.7.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.

Possible Problems:
- None
Test # 1.8: Extra Trailing Octets

**Purpose:** This tests the DUT’s ability to identify and accept requests that have extra octets.

**References:**
[1] Section 3.1.1.8 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** August 15, 2006

**Discussion:**
This is a well-formed request and should be accepted as such. The extra octet looks like an INVITE request but is just extra noise that should not be processed. If this message went through a proxy, only the REGISTER request should be forwarded.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send the REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 1.8.

**Observable Results:**
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that the false INVITE octet was ignored and not processed as a real invite.
3. If the device is a proxy, verify that the DUT remove the false INVITE octet.

**Possible Problems:**
- None
Test # 1.9: Semicolon Separated Parameters in the URI Userpart

Purpose: This tests the DUT’s ability to recognize and accept a URI that has a semicolon separating the parameters.

References:
[1] Section 3.1.1.9 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion: This is a well-formed request and should be accepted as such. This request contains an escaped “@” symbol which should parse out normally as an appropriate address.

Test Layout:
- Test Setup 1

Procedure:
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 1.9.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that the escaped values were incorporated as part of the address.

Possible Problems:
- None
Test # 1.10: Unknown Transport Types

**Purpose:** This tests the DUT’s ability to recognize and accept requests containing unknown or varied transport types.

**References:**
- [1] Section 3.1.1.10 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 13, 2006

**Discussion:**
This is a well-formed request and should be accepted as such. This request should be processed exactly as if the 2nd, 3rd, and 4th header fields specified UDP.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send the OPTIONS from the SIP Endpoint to the DUT as defined by Test Message 1.10.

**Observable Results:**
1. Verify that the DUT responded with a “200 Ok”.

**Possible Problems:**
- None
Test # 1.11: Multipart MIME Messages

Purpose: This tests the DUT’s ability to identify and accept requests that have multiple “body” sections.

References:
[1] Section 3.1.1.11 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion: This is a well-formed request and should be accepted as such. Even if this multipart form is unsupported by the DUT. This request contains two “body” sections. The 2nd section is in binary and contains null values. It is important to note that this message must be framed properly.

Test Layout:
- Test Setup 1

Procedure:
1. Send the REGISTER from the SIP Endpoint to the DUT as defined by Test Message 1.11.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that both parts of the body were received and framed properly.

Possible Problems:
- None
Test # 1.12: Unusual Reason Phrase

Purpose: This tests the DUT’s ability to identify and accept requests that have an atypical reason phrase.

References:
[1] Section 3.1.1.12 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion:
This is a well-formed response and should be accepted as such. This 200 response contains a reason phrase other than “ok”. This reason phrase is considered for human consumption only and should be ignored in processing. It is also important to note that this response has unusual characters, however the entire response is valid.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the DUT to the SIP Endpoint.
2. Send a “200 Ok” response from the SIP Endpoint to the DUT as defined by Test Message 1.12.

Observable Results:
1. Verify that the DUT responded with an “ACK” to the “200 Ok”.

Possible Problems:
- None
Test # 1.13: No Reason Phrase

Purpose: This tests the DUT's ability to identify and accept requests that have no reason phrase.

References:
[1] Section 3.1.1.13 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 13, 2006

Discussion:
This is a well-formed response and should be accepted as such. This response has no reason phrase. As described the previous test, reason phrases are intended only for human consumption, and are not necessary for the response to be processed.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE from the DUT to the SIP Endpoint.
2. Send the response from the SIP Endpoint to the DUT as defined by Test Message 1.13.

Observable Results:
1. Verify that the DUT responded with an “ACK” to the response.

Possible Problems:
- None
Group 2: Invalid Messages

Scope:
Tests within this group focus on how a DUT should reject invalid requests and responses.

Overview:
The following tests assess how a DUT should respond to a series of invalid messages that should be rejected. These invalid messages include excessive punctuation in headers, invalid content length, excessive values, and incorrect quotation.
Test # 2.1: Extraneous Header Field Separators

Purpose: This test verifies that the DUT identifies and rejects excessive punctuation in field headers.

References:
[1] Section 3.1.2.1 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 14, 2006

Discussion:
This INVITE contains excessive semicolons and commas after several of the header fields. This punctuation has no values or parameters, making the message invalid and it should be rejected as such.

Test Layout:
- Test Setup 1

Procedure:
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.1.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.

Possible Problems:
- None
Test # 2.2: Content Length Larger than the Message

Purpose: This test verifies that the DUT can identify and reject messages with content length larger than the actual message size.

References:
[1] Section 3.1.2.2 (RFC 4475)

Resource Requirements:
• SIP Endpoint
• DUT

Last Modification: August 7, 2006

Discussion:
Based on RFC 4475, when an element receives a request message containing a Content-Length header value larger than the true size of the message, it should respond with a simple “400 Bad Request” error (if using an unreliable transport, such as UDP). However, over a TCP or other reliable transport connection the DUT should wait for more information to be sent, and close the connection if none is received in a reasonable period of time.

Test Layout:
• Test Setup 1

Procedure:
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.2.

Observable Results:
1. If UDP, Verify that the DUT responded with a “400 Bad Request”.
2. If TCP, Verify the DUT closed the session in a reasonable amount of time.

Possible Problems:
• None
Test # 2.3: Negative Content Length

**Purpose:** This test verifies that the DUT can identify and reject requests with and invalid content length.

**References:**
[1] Section 3.1.2.3 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** August 7, 2006

**Discussion:**
According to RFC 4475 should the content length header contain an invalid value, such as a negative value or any non-numeric character, the rest of the message should be discarded. Any connection should be closed, as this is a non-recoverable error.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Issue an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.3.

**Observable Results:**
1. If UDP, verify that DUT responded with a “400 Bad Request”.
2. If TCP, verify that DUT closed the session.

**Possible Problems:**
- None
Test # 2.4: Request Scalar Fields with Overlarge Values

**Purpose:** This test verifies that the DUT can identify and reject requests with scalar values outside of their legal range.

**References:**
[1] Section 3.1.2.4 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 17, 2006

**Discussion:**
This request contains a CSeq number, a Max Forward value, an Expire value, and a Contact Expires value that are outside of their legal range. If Max Forward is the only illegal value then the DUT should process the message as if that field were absent. If the Expire value was the only illegal values then the DUT should treat them as if they contained the default values (in this case 3600).

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 2.4.

**Observable Results:**
1. Verify that the DUT responded with a “400 Bad Request”.

**Possible Problems:**
- None

**Note:** This test could be expanded to test several different fields, as a few fields, described in the discussion, can be responded to without a 400 error. The Contact q value, the Timestamp value, and the Via ttl parameter could also be included in this test.

**Important Note:** This Test has been moved to sections 2.5-2.9 in the Conformance Test Suite.
Test # 2.5: Response Scalar Fields with Overlarge Values

**Purpose:** This test verifies that the DUT can identify and reject responses with scalar values outside their legal range.

**References:**
[1] Section 3.1.2.5 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** August 7, 2006

**Discussion:**
This response contains a CSeq value and a Warning field value with illegal values. It is important to note that while the Retry After field has no defined legal range, in this example it is unreasonably large and should be rejected.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send an OPTIONS request from the DUT to the SIP Endpoint.
2. Send a 503 response from the SIP Endpoint to the DUT as defined by Test Message 2.5.

**Observable Results:**
1. Verify that the DUT discarded this response.

**Possible Problems:**
- It may not be possible to cause the DUT to send the required OPTIONS request.
- It may not be possible to verify the DUT discards the invalid response.
Test # 2.6: Unterminated Quoted String

**Purpose:** This test verifies that the DUT can identify and reject requests with an unended quote in the Display Header.

**References:**
[1] Section 3.1.2.6 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 17, 2006

**Discussion:**
In most cases the DUT should respond to this request with a “400 Bad Request”. However, it is legal for the DUT to attempt to infer an ending quote and accept this request. Care needs to be taken to verify that the inference was correct.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.6.

**Observable Results:**
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify if it correctly inferred a terminating quote to the header.

**Possible Problems:**
- None
Test # 2.7: “<>” Enclosing the URI

Purpose: This test verifies that the DUT can identify and reject requests in which the request-URI is enclosed in “<>”.

References:
[1] Section 3.1.2.7 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion:
It is always reasonable to reject this request with a 400 error. However some liberal DUTs may accept this request by ignoring the brackets. If the DUT is forwarding this message, it should remove the brackets from the message.

Test Layout:
- Test Setup 1

Procedure:
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.7.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify if it correctly ignored the bracket.
3. If the DUT forwards the request, verify it remove the brackets from the message.

Possible Problems:
- None
Test # 2.8: Malformed SIP Request URI

Purpose: This test verifies that the DUT can identify and reject requests with an illegal linear white space in the URI.

References:
[1] Section 3.1.2.8 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 17, 2006

Discussion:
This request contains an illegal linear white space in its URI. It is always acceptable to respond with a 400 error. However, a liberal DUT could attempt to ignore the linear white space so long as the resulting message is well formed.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.8.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify it ignored the linear white space.

Possible Problems:
- None
Test # 2.9: Multiple SP Separating Request Line Elements

Purpose: This test verifies that the DUT can identify and reject illegal spaces.

References:
[1] Section 3.1.2.9 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion:
This request contains extra spaces in the Request-URI, and which can always be rejected as malformed. A liberal DUT may choose instead to ignore these extra spaces if the resulting message is well formed. Any forwarding by the DUT should remove these extra spaces.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.9.

Observable Results:
1. Verify that the DUT responded the request with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify if it correctly ignored the extra spaces.
3. If forwarded, verify that the DUT removed the extra spaces.

Possible Problems:
- None
Test # 2.10: SP Characters at the End of the Request Line

**Purpose:** This test verifies that the DUT can identify and reject requests with extra spaces in the request line URI.

**References:**

[1] Section 3.1.2.10 (RFC 4475)

**Resource Requirements:**

- SIP Endpoint
- DUT

**Last Modification:** August 7, 2006

**Discussion:**
This request contains extra spaces after the "protocol-version" parameter of the Request-Line. As above these can be ignored and/or removed by the DUT and should be removed if this request is forwarded.

**Test Layout:**

- Test Setup 1

**Procedure:**

1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.10.

**Observable Results:**

1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify if it correctly ignored the spaces.
3. If forwarded, verify that the DUT removed the extra spaces.

**Possible Problems:**

- None
Test # 2.11: Escaped Headers (%) in the Request URI

**Purpose:** This test verifies that the DUT can identify and reject illegal “%” escapes in the URI.

**References:**
- [1] Section 3.1.2.11 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 17, 2006

**Discussion:**
“%” escapes are not equivalent to the values they represent when put into field headers and URIs. This INVITE is an example of an illegal use of escapes and should be rejected by the DUT. It is also possible that the DUT could ignore the escaped header; however, if this message was forwarded then the escaped values should not be translated.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.11.

**Observable Results:**
1. Verify that the DUT responded with a “400 Bad Request”
2. If forwarded, verify that the message was not changed and the “%” escaped values are not translated.

**Possible Problems:**
- None
Test # 2.12: Invalid Time Zone

Purpose: This test verifies that the DUT can identify and reject requests with incorrect time zones.

References:
[1] Section 3.1.2.12 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 17, 2006

Discussion:
The only acceptable time zone for SIP is GMT as defined by RFC 3261. It is also important to note that UT, UTC and UCT while synonymous to GMT are not valid time zones for the Time Zone field. As we can see in the date header field, GMT was not used therefore this request is malformed and can be rejected as such. A liberal DUT could ignore the date field all together.

Test Layout:
- Test Setup 1

Procedure:
1. Send the INVITE request from the SIP Endpoint to the DUT as defined by Test Message 2.12.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.

Possible Problems:
- None
Test # 2.13: Name-Addr Format URI without “<>”

Purpose: This test verifies that the DUT can identify and reject requests without name-addr form URIs enclosed in “<>”.

References:
[1] Section 3.1.2.13 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
Under the contact field the URI is using “%” escapes. This requires that “<>” be used to enclose the name as per the name-addr format. It is always legal to respond to this request with a 400. In this case, because of the lack of ambiguity, a liberal DUT could infer “<>”. In most cases however, this should not occur.

Test Layout:
- Test Setup 1

Procedure:
1. Send the REGISTER from the SIP Endpoint to the DUT as defined by Test Message 2.13.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify that the brackets were correctly inferred.

Possible Problems:
- None
Test # 2.14: Spaces Within Addr-Spec

Purpose: This test verifies that the DUT can identify and reject requests with illegal spaces in the addr-spec.

References:
[1] Section 3.1.2.14 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
There are illegal spaces in the To header for this request. It is always acceptable to respond with a 400 error. A DUT that is liberal may attempt to ignore the spaces.

Test Layout:
- Test Setup 1

Procedure:
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.14.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify that the spaces were correctly ignored.

Possible Problems:
- None
Test # 2.15: Non-token Characters in the Display Name

**Purpose:** This test verifies that the DUT can identify and reject requests with unquoted, non-token characters in the Display name header.

**References:**
- [1] Section 3.1.2.15 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 18, 2006

**Discussion:**
In a display name all non-token characters should be quoted. This is not the case in the To and From fields. It is always acceptable to reject this request with a 400, and if the DUT is a proxy, this request should always be rejected. If the DUT is a liberal endpoint, then it may attempt to infer the brackets.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.15.

**Observable Results:**
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the DUT did not respond with an error, verify that the quotes were correctly inferred.

**Possible Problems:**
- None
Test # 2.16: Unknown Protocol Version

Purpose: This test verifies that the DUT can identify and reject requests with an unknown version number.

References:
[1] Section 3.1.2.16 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
This request has an unknown version number. It should always be rejected as malformed and never forwarded.

Test Layout:
- Test Setup 1

Procedure:
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.16.

Observable Results:
1. Verify that the DUT responded with a “505 Version Not Supported”.

Possible Problems:
- None
Test # 2.17: Request Line and CSeq Method Mismatch

Purpose: This test verifies that the DUT can identify and reject requests with different methods in 2 separate fields.

References:
[1] Section 3.1.2.17 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
This request has 2 separate methods listed in the CSeq field and the request line, in this case OPTIONS and INVITE. This is a malformed request and should be rejected.

Test Layout:
- Test Setup 1

Procedure:
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.17.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.

Possible Problems:
- None
Test # 2.18: Unknown Method with CSeq Method Mismatch

**Purpose:** This test verifies that the DUT can identify and reject requests with an unknown method and a known method listed in 2 separate fields.

**References:**
[1] Section 3.1.2.18 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 18, 2006

**Discussion:**
As with the test above, this is a malformed request and should be rejected. However, instead of responding with a 400 error, a 501 error is preferable in this case, particularly if the DUT is a proxy.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send the OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 2.18.

**Observable Results:**
1. Verify that the DUT responded with a “501 Not Implemented”.
2. If not, verify that the DUT responded with a “400 Bad Request”.

**Possible Problems:**
- None
Test # 2.19: Overlarge Response Code

Purpose: This test verifies that the DUT can identify and reject responses with response codes that are too large.

References:
[1] Section 3.1.2.19 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 15, 2006

Discussion:
This response has a response code that is larger than 699, the limit for this value. This is a malformed and should be dropped.

Test Layout:
- Test Setup 1

Procedure:
1. Send the INVITE request from the DUT to the SIP Endpoint.
2. Send a response from the SIP Endpoint to the DUT as defined by Test Message 2.19.

Observable Results:
1. Verify that the DUT dropped this response.

Possible Problems:
- The DUT may respond with a 400 error because this is an invalid packet.
Group 3: Transaction Layer Semantics

Scope:
The following tests verify the logic that a DUT is expected to have for the transaction layer.

Overview:
The following tests check to ensure that all transaction identifiers are in place.
Test # 3.1: Missing Transaction Identifiers

Purpose: This test verifies that the DUT is able to function when no RFC 3261 transaction identifiers are present.

References:
[1] Section 3.2.1 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 15, 2006

Discussion:
This request indicates it supports RFC 3261 compatible identifiers, but provides none. Any DUT to receive this must not break. A DUT could either respond with a 400 error or fall back to RC 2543 compatible transaction identifiers.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 3.1.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. Verify that the DUT can support a valid call after a period of about 2 minutes.

Possible Problems:
- It is important to note that there may be other responses then just a “400 Bad Request” that are acceptable.
Group 4: Application Layer Semantics

Scope:
The following tests verify the logic that a DUT is expected to have for the application layer.

Overview:
The following test checks to the DUT to ensure it will function properly when receiving requests containing unknown methods and headers. It also checks that the DUT responds properly to requests when they are missing critical header fields.
Test # 4.1: Missing Required Header Fields

Purpose: This test assesses the DUT’s ability to identify and reject requests without critical header fields.

References:
[1] Section 3.3.1 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
This request is missing several key header fields including To, From, and Call-ID. The DUT must not break, but should reject this request as malformed and respond with a 400 error.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 4.1.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.

Possible Problems:
- None
Test # 4.2: Request URI with Unknown Scheme

Purpose: This test assesses the DUT's ability to identify and reject requests with an unknown URI scheme in the request line.

References:
[1] Section 3.3.2 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
This request doesn’t contain a recognizable URI scheme in the request line. This is a well-formed SIP request, but it should be rejected nonetheless with a 416-error message. It is important to note that some earlier versions of SIP might have tried to look at the To header field in order to salvage this error, but this is not a valid way of fixing this error.

Test Layout:
- Test Setup 1

Procedure:
1. Send an OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 4.2.

Observable Results:
1. Verify that the DUT responded with a “416 Unsupported URI Scheme”.

Possible Problems:
- None
Test # 4.3: Request URI with Known But Atypical Scheme

Purpose: This test assesses the DUT’s ability to identify and reject requests without critical header fields.

References:
[1] Section 3.3.3 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion:
This request contains a URI with an IANA-registered scheme that does normally not appear in SIP but is still valid. If the DUT will never accept this URI scheme, it should send a 416 error. If the DUT can accept this scheme, it should send a 404 error for those schemes it doesn’t accept.

Test Layout:
- Test Setup 1

Procedure:
1. Send an OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 4.3.

Observable Results:
1. Verify that the DUT responded with a “404 Not Found” or a “416 Unsupported URI Scheme” or a “200 Ok” depending on the DUT’s support of IANA-registered schemes.

Possible Problems:
- None
Test # 4.4: Unknown URI in Header Scheme

Purpose: This test assesses the DUT’s ability to identify and reject requests with unknown schemes in key header fields.

References:
[1] Section 3.3.4 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion: This request has unknown schemes several key header fields including To, From, and Contact. The DUT must not break, or fail when receiving this, as it is syntactically valid. A proxy should forward this as normal but an endpoint or registrar should reject this request and respond with a 400 error.

Test Layout:
- Test Setup 1

Procedure:
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by
2. 
3. Test Message 4.4

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.
2. If the device was a proxy, verify that the DUT forwarded this response as normal.

Possible Problems:
- None
Test # 4.5: Unsupported Requires and Proxy-Requires

**Purpose:** This test assesses the DUT’s ability to identify and reject requests with unsupported requirements.

**References:**
[1] Section 3.3.5 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** August 7, 2006

**Discussion:**
This request has unsupported values for the Require and Proxy-Require header fields. A DUT receiving this request will respond with a 420 and list in the unknown heading any extensions it didn’t support.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send an OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 4.5.

**Observable Results:**
1. Verify that the DUT responded with a “420 Bad Extension” and lists the bad extension.

**Possible Problems:**
- None
Test # 4.6: Unknown Content Type

Purpose: This test assesses the DUT’s ability to identify and reject requests with unsupported content in the body of the message.

References:
[1] Section 3.3.6 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion: This request is syntactically valid, so any DUT receiving it must not fail, however, it contains a media type that is not supported. A proxy should forward this as normal, however, an endpoint would have to respond with a 415.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 4.6.

Observable Results:
1. Verify that the DUT responded with a “415 Unsupported Media Type”.
2. If the device was a proxy, verify the DUT forwarded this request as normal.

Possible Problems:
- None
Test # 4.7: Unknown Authorization Scheme

Purpose: This test assesses the DUT’s ability to identify and reject requests with unsupported content in the body of the message.

References:
[1] Section 3.3.7 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion:
This request is well formed but contains an authentication scheme that is unknown. A proxy should treat this request like any other register and should forward the authorization field unmodified. An endpoint should ignore the authorization header unless it enforces a challenge-response in which case it will give a 401 followed by a challenge in a scheme it understands. An endpoint not behaving like a registrar will respond with a 405.

Test Layout:
- Test Setup 1

Procedure:
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 4.7.

Observable Results:
1. Verify that the DUT responded with a “401 Unauthorized”.
2. If it is not acting as a registrar, respond with a “405 Method Not Allowed”.
3. If the device was a proxy, verify the request was forwarded unmodified.

Possible Problems:
- None
Test # 4.8: Multiple Values in Single Value Fields

Purpose: This test assesses the DUT’s ability to identify and reject requests with multiple values in a single value field.

References:
[1] Section 3.3.8 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 7, 2006

Discussion:
This request contains several header fields such as Call-ID, To and From that occur multiple times. A DUT receiving this request must not break. This request is invalid, and should be responded to with a 400-error message.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 4.8.

Observable Results:
1. Verify that the DUT responded with a “400 Bad Request”.

Possible Problems:
- None
Test # 4.9: Multiple Content Length Values

**Purpose:** This test assesses the DUT’s ability to identify and reject requests with multiple content length headers.

**References:**
[1] Section 3.3.9 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** July 18, 2006

**Discussion:**
A valid SIP message may only contain one content length value. This request has multiple values and therefore all of them must be ignored, as there are conflicting numbers as to how many octets there are.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send an OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 4.9.

**Observable Results:**
1. Verify that the DUT responded with a “400 Bad Request”.

**Possible Problems:**
- None
Test # 4.10: Response with Broadcast Via Header Field Value

Purpose: This test assesses the DUT’s ability to reject responses with a broadcasting address (255.255.255.255) in the Via headers.

References:
[1] Section 3.3.10 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: August 15, 2006

Discussion:
This response is a well-formed 200 Ok response. If a DUT was blindly following the standard it would forward this to the address listed, however because this is a possible avenue for attack this response should just be discarded.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the DUT to the SIP Endpoint.
2. Send a response from the SIP Endpoint to the DUT as defined by Test Message 4.10.

Observable Results:
1. Verify that the DUT dropped this response.

Possible Problems:
- None
Test # 4.11: Max-Forward of Zero

Purpose: This test assesses the DUT’s ability to identify and reject requests with a Max-Forward Value of zero.

References:
[1] Section 3.3.11 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 18, 2006

Discussion:
This is a well-formed and legal request, however the max-forward value is zero, meaning that this message can no longer be sent. A proxy should respond with a 483. An endpoint should process it normally.

Test Layout:
- Test Setup 1

Procedure:
1. Send an OPTIONS request from the SIP Endpoint to the DUT as defined by Test Message 4.11.

Observable Results:
1. If the device was a Proxy verify that it responded with a “483 To Many Hops”
2. If the device was an Endpoint, verify that it correctly accepted the request with a “200 Ok”.

Possible Problems:
- None
Test # 4.12: REGISTER with a Contact Header Parameter

**Purpose:** This test assesses the DUT's ability to identify and accept requests with unusual parameters in the contact information.

**References:**
[1] Section 3.3.12 (RFC 4475)

**Resource Requirements:**
- SIP Endpoint
- DUT

**Last Modification:** August 7, 2006

**Discussion:**
This is a well-formed and valid request and should be responded to normally. It is important to note that the contact parameter name is not equivalent to the URL parameter. Therefore the DUT cannot use the contact parameter (in this case “unknownparam”) as the URL parameter.

**Test Layout:**
- Test Setup 1

**Procedure:**
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 4.12.

**Observable Results:**
1. Verify that the DUT responded with a “200 Ok”.
2. Verify that the DUT did not use the contact parameter (unknownparam) for its URL parameter in the response.

**Possible Problems:**
- None
Test # 4.13: REGISTER with a URL Parameter

**Purpose:** This test assesses the DUT’s ability to identify and accept requests with unusual parameters in the URL.

**References:**

[1] Section 3.3.13 (RFC 4475)

**Resource Requirements:**

- SIP Endpoint
- DUT

**Last Modification:** July 19, 2006

**Discussion:**
The URL in this request has an unknown parameter. This should be accepted, as it is well formed and valid. Any retrieval of this registration should get an “unkownparam” as the URL parameter.

**Test Layout:**

- Test Setup 1

**Procedure:**

1. Send a REGISTER request from the DUT to the SIP Endpoint as defined by Test Message 4.13.

**Observable Results:**

1. Verify that the DUT responded with a “200 Ok”.

**Possible Problems:**

- None
Test # 4.14: REGISTER with a URL Escaped Header

Purpose: This test assesses the DUT’s ability to identify and accept requests with escaped headers.

References:
[1] Section 3.3.14 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 19, 2006

Discussion:
The URL in this request has an escaped header in the contact field. This is a valid and well-formed message and should be accepted as such. Any retrieval of this register must include the escaped route header in the contact field.

Test Layout:
- Test Setup 1

Procedure:
1. Send a REGISTER request from the SIP Endpoint to the DUT as defined by Test Message 4.14.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.

Possible Problems:
- None
Test # 4.15: Unacceptable Accept Offering

Purpose: This test assesses the DUT’s ability to identify and reject requests with an invalid Accept header.

References:  
[1] Section 3.3.15 (RFC 4475)

Resource Requirements:  
- SIP Endpoint  
- DUT

Last Modification: August 7, 2006

Discussion:  
In this request the Accept header contains an “unknown”, indicating that the body of this message should be in an unknown format. A DUT receiving this should respond ideally with a 406 followed by a Warning/399 indicating that it cannot formulate the requests based on the formats in the accept field. It is also acceptable to respond with a 400 error, though this is less helpful.

Test Layout:  
- Test Setup 1

Procedure:  
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 4.15.

Observable Results:  
1. Verify that the DUT responded with a “406 Not Acceptable” or a “400 Bad Request”

Possible Problems:  
- The DUT could also include a 399 warning with the 406 for human consumption.
Group 5: Backwards Compatibility

Scope:
The Following test verifies a DUT’s ability to be compatible with older standards.

Overview:
The following tests goes over several legal differences between RFC 3261 and previous standards.
Test # 5.1: INVITE with RFC 2543 Syntax

Purpose: This test verifies that the DUT can verify and accept invites with legal usage from the previous standard RFC 2543.

References:
[1] Section 3.4.1 (RFC 4475)

Resource Requirements:
- SIP Endpoint
- DUT

Last Modification: July 19, 2006

Discussion:
This is a legal request that should be properly accepted by DUT’s with backwards compatibility. There are four primary things to note in this older format. There is no branch parameter in the Via header. There is no from tag. There is no explicitly defined content-length. Lastly, there is no Max-Forward header.

Test Layout:
- Test Setup 1

Procedure:
1. Send an INVITE request from the SIP Endpoint to the DUT as defined by Test Message 5.1.

Observable Results:
1. Verify that the DUT responded with a “200 Ok”.

Possible Problems:
- None
Appendix 1: Test Messages

Table of Contents

For accuracy’s sake, the examples used by RFC 4475 for complex grammar have been copied here exactly as they are seen in RFC 4475. Due to the text-based constraints seen in RFCs, there are a few unusual abbreviations and space saving functions that have been copied over as well. Here is the explanation.

Representing Long Lines

Several of these examples contain unfolded lines longer than 72 characters. These are captured between <allOneLine/> tags. The single unfolded line is reconstructed by directly concatenating all lines appearing between the tags (discarding any line feeds or carriage returns). There will be no whitespace at the end of lines. Any whitespace appearing at a fold-point will appear at the beginning of a line.

The following represent the same string of bits:

    Header-name: first value, reallylongsecondvalue, third value

    <allOneLine>
    Header-name: first value, reallylongsecondvalue , third value
    </allOneLine>

    <allOneLine>
    Header-name: first value, reallylong second value, third value
    </allOneLine>

Note that this is NOT SIP header-line folding, where different strings of bits have equivalent meaning.

Representing Non-printable Characters

Several examples contain binary message bodies or header field values containing non-ascii range UTF-8 encoded characters. These are rendered here as a pair of hexadecimal digits per octet between <hex/> tags. This rendering applies even inside quoted-strings.

The following represent the same string of bits:

    Header-name: value one
    Header-name: value<hex>206F6E</hex>e

The following is a Subject header field containing the euro symbol:

    Subject: <hex>E282AC</hex>

2.3. Representing Long Repeating Strings

Several examples contain very large data values created with repeating bit strings. Those will be rendered here using <repeat count=some_integer>value</repeat>. As with <hex>, this rendering applies even inside quoted strings.
For example, the value "abcabcabc" can be rendered as \(<\text{repeat count=3}>abc</\text{repeat}>\). A display name of "1000000 bottles of beer" could be rendered as

To: \"1<\text{repeat count=6}><\text{hex>30</hex>}</\text{repeat}> bottles of beer\"<\text{sip:beer.example.com}>

A Max-Forwards header field with a value of one google will be rendered here as

Max-Forwards: 1<\text{repeat count=100}>0</\text{repeat}>

Test Message 1.1

Test # 1.1: Accepting a Torturous INVITE

Message Details : wsinv

INVITE sip:vivekg@chair-dnrc.example.com:unknownparam SIP/2.0
TO : sip:vivekg@chair-dnrc.example.com ; tag  = 1918181833n
from : "J Rosenberg \"\" <sip:jdrosen@example.com>
 ;
 ;
tag = 98asjd8
MaX-fOrWaRdS: 0068
Call-ID: wsinv.ndaksdj@192.0.2.1
Content-Length : 150
cseq: 0009
INVITE
Via : SIP / 2.0
/UDP
192.0.2.2;branch=390skduw
s :
NewFangledHeader: newfangled value
continued newfangled value
UnknownHeaderWithUnusualValue: ;;,;;
Content-Type: application/sdp
Route:
<sip:services.example.com;lr;unknownwith=value;unknown-no-value>
v: SIP / 2.0 / TCP spindle.example.com ;
branch = z9hG4bK9ikj8 ,
SIP / 2.0 / UDP 192.168.255.111 ; branch=value
z9hG4bK30239
m:"Quoted string \"\" <sip:jdrosen@example.com> ;
newparam =

newvalue ;
secondparam ; q = 0.33

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.3
s=-
c=IN IP4 192.0.2.4
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC
Test Message 1.2
Test # 1.2: Accepting Valid Characters
Message Details: intmeth

<allOneLine>
!interesting-Method0123456789_+`.%indeed~
sip:1_unusual.URI~(to-be!sure)&isn't+it$/crazy?,/;,*
:&it+=1,weird!*pass$wo-d_too.(doesn't-it)
@example.com SIP/2.0
</allOneLine>
Via: SIP/2.0/TCP host1.example.com;branch=z9hG4bK-.!%66*+`~
<allOneLine>
To: "BEL:\<hex>07</hex> NUL:\<hex>00</hex> DEL:\<hex>7F</hex>"
  <sip:1_unusual.URI~(to-be!sure)&isn't+it$/crazy?,/;,*
@example.com>
</allOneLine>
From: token1~` token2'+_ token3*%!.- <sip:mundane@example.com
;fromParam''~+*_!.-%=
"<hex>D180D0B0D0B1D0BED182D0B0D18ED189D0B8D0B9</hex>
;tag=_token~1'+`*%!-
</allOneLine>
Call-ID: intmeth.word%ZK-!.*_+'@word`~)(><:/"\[?}{
CSeq: 139122385 !interesting-Method0123456789_+`.%indeed~
Max-Forwards: 255
<allOneLine>
extensionHeader-.!%*+`~:
<hex>EFBBBFE5A4A7E5819CE99BBB</hex>
</allOneLine>
Content-Length: 0

Test Message 1.3
Test # 1.3: Valid use of “%” as an Escape
Message Details: esc01

INVITE sip:sips%3Auser%40example.com@example.net SIP/2.0
To: sip:%75se%72@example.com
From: <sip:I%20have%20spaces@example.net
;tag=938
Max-Forwards: 87
i: esc01.239409asdfakjkn23onasd0-3234
CSeq: 234234 INVITE
Via: SIP/2.0/UDP host5.example.net;branch=z9hG4bKkdjiuw
C: application/sdp
Contact:
  <sip:cal%6Cer@host5.example.net;%6C%72;n%61me=v%61lue%25%34%31>
Content-Length: 150

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.1
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC
Test Message 1.4
Test # 1.4: Escaped Nulls in URIs
Message Details : escnull
REGISTER sip:example.com SIP/2.0
To: sip:null-%00-null@example.com
From: sip:null-%00-null@example.com;tag=839923423
Max-Forwards: 70
Call-ID: escnull.39203ndfvdasfkdq3w4trq0adsfndnavd
CSeq: 14398234 REGISTER
Via: SIP/2.0/UDP host5.example.com;branch=z9hG4bKkdjuw
Contact: <sip:%00@host5.example.com>
Contact: <sip:%00%00@host5.example.com>
L: 0

Test Message 1.5
Test # 1.5: Valid Use of “%” When Not as an Escape
Message Details : esc02
RE%47IST%45R sip:registrar.example.com SIP/2.0
To: "%Z%45" <sip:resource@example.com>
From: "%Z%45" <sip:resource@example.com>;tag=f232jadfj23
Call-ID: esc02.asdfnqwo34rq23i34jjasdcn23nrlksdf
Via: SIP/2.0/TCP host.example.com;branch=z9hG4bK209%fzsnel234
CSeq: 29344 RE%47IST%45R
Max-Forwards: 70
Contact: <sip:alias1@host1.example.com>
Contact: <sip:alias2@host2.example.com>
Contact: <sip:alias3@host3.example.com>
L: 0

Test Message 1.6
Test # 1.6: Message with No Linear White Space in the Display Name
Message Details : lwsdisp

OPTIONS sip: user@example.com SIP/2.0
To: sip: user@example.com
From: caller <sip: caller@example.com>; tag=323
Max-Forwards: 70
Call-ID: lwsdisp.1234abcd@funky.example.com
CSeq: 60 OPTIONS
Via: SIP/2.0/UDP funky.example.com; branch=z9hG4bKkdjuw
L: 0

Test Message 1.7
Test # 1.7: Long values in Header Fields
Message Details : longreq

INVITE sip: user@example.com SIP/2.0
<allOneLine>
To: "I have a user name of
<repeat count=10>extreme</repeat> proportion"
<sip: user@example.com>;6000;
unknownparam1=very<repeat count=20>long</repeat>value;
longparam<repeat count=25>name</repeat>=shortvalue;
very<repeat count=25>long</repeat>ParameterNameWithNoValue>
</allOneLine>
<allOneLine>
F: sip: <repeat count=5>amazinglylongcallername</repeat>@example.net
;tag=12<repeat count=50>424</repeat>
;unknownheaderparam<repeat count=20>name</repeat>=
unknownheaderparam<repeat count=15>value</repeat>
;unknownValueless<repeat count=10>paramname</repeat>
</allOneLine>
Call-ID: longreq.one<repeat count=20>really</repeat>longcallid
CSeq: 3882340 INVITE
<allOneLine>
Unknown-<repeat count=20>Long</repeat>-Name:
unknown-<repeat count=20>long</repeat>-value;
unknown-<repeat count=20>long</repeat>-parameter-name =
unknown-<repeat count=20>long</repeat>-parameter-value
</allOneLine>
Via: SIP/2.0/TCP sip33.example.com
v: SIP/2.0/TCP sip32.example.com
V: SIP/2.0/TCP sip31.example.com
Via: SIP/2.0/TCP sip30.example.com
ViA: SIP/2.0/TCP sip29.example.com
V1a: SIP/2.0/TCP sip28.example.com
VIA: SIP/2.0/TCP sip27.example.com
via: SIP/2.0/TCP sip26.example.com
viA: SIP/2.0/TCP sip25.example.com
v1a: SIP/2.0/TCP sip24.example.com
v1A: SIP/2.0/TCP sip23.example.com
V : SIP/2.0/TCP sip22.example.com
v : SIP/2.0/TCP sip21.example.com
V : SIP/2.0/TCP sip20.example.com
Test Message 1.8

Test # 1.8: Extra Trailing Octets

Message Details : dblreq

REGISTER sip:example.com SIP/2.0
To: sip:j.user@example.com
From: sip:j.user@example.com;tag=43251j3j324
Max-Forwards: 8
I: dblreq.0ha0isndaksdj99sdfafnl3lk233412
Contact: sip:j.user@host.example.com
CSeq: 8 REGISTER
Via: SIP/2.0/UDP 192.0.2.125;branch=z9hG4bKkdjuw23492
Content-Length: 0
INVITE sip:joe@example.com SIP/2.0
t: sip:joe@example.com
From: sip:caller@example.net;tag=141334
Max-Forwards: 8
Call-ID: dbreq.0ha0isnda977644900765@192.0.2.15
CSeq: 8 INVITE
Via: SIP/2.0/UDP 192.0.2.15;branch=z9hG4bKkdjuw380234
Content-Type: application/sdp
Content-Length: 150

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.15
s=-
c=IN IP4 192.0.2.15
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 1.9
Test # 1.9: Semicolon Separated Parameters in the URI Userpart
Message Details : semiuri

OPTIONS sip:u%40example.net@example.com SIP/2.0
To: sip:j_user@example.com
From: sip:caller@example.org;tag=33242
Max-Forwards: 3
Call-ID: semiuri.0ha0isndaksdj
CSeq: 8 OPTIONS
Accept: application/sdp, application/pkcs7-mime,
multipart/mixed, multipartSigned,
message/sip, message/sipfrag
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bKkdjuw
l: 0

Test Message 1.10
Test # 1.10: Unknown Transport Types
Message Details : transports

OPTIONS sip:user@example.com SIP/2.0
To: sip:user@example.com
From: <sip:caller@example.com>;tag=323
Max-Forwards: 70
Call-ID: transports.kijh4akdnaqkfwendasfdj
Accept: application/sdp
CSeq: 60 OPTIONS
Via: SIP/2.0/UDP t1.example.com;branch=z9hG4bKkdjuw
Via: SIP/2.0/SCTP t2.example.com;branch=z9hG4bKklasdjhf
Via: SIP/2.0/TLS t3.example.com;branch=z9hG4bK2980unddj
Via: SIP/2.0/UNKNOWN t4.example.com;branch=z9hG4bKasd0f3en
Via: SIP/2.0/TCP t5.example.com;branch=z9hG4bK0a9idfnee
Test Message 1.11
Test # 1.11: Multipart MIME Messages
Message Details : mpart01

MESSAGE sip:kumiko@example.org SIP/2.0
Via: SIP/2.0/UDP 127.0.0.1:5070
;branch=z9hG4bK-d87543-4dade06d0db11ee-1--d87543--rport
Max-Forwards: 70
Route: <sip:127.0.0.1:5080>
Identity: r5mwreLuyDRYBi/0TiPwEsY3rEVsk/G2WxhgTV1PF7hHuL
lK0YWKZhKv9Mj8UeXqkMVbnVq37CD+813gyYjcBUaZngQmx9WNNZSDN
GCzA+fWf9mEUHfWZo1CeJebdYfXlgKeTaOlvq0r70Q5jiSfbqMJmQF
teeivUkhMWYUA=
Contact: <sip:fluffy@127.0.0.1:5070>
To: <sip:kumiko@example.org>
From: <sip:fluffy@example.com>;tag=2fb0dce9
Call-ID: 3d9485ad0c49859b@Zmx1ZmZ5LW1hYy0xNi5sb2NhA..
CSeq: 1 MESSAGE
Content-Transfer-Encoding: binary
Content-Type: multipart/mixed;boundary=7a9cbec02ceef655
Date: Sat, 15 Oct 2005 04:44:56 GMT
User-Agent: SIPimp.org/0.2.5 (curses)
Content-Length: 553

--7a9cbec02ceef655
Content-Type: text/plain
Content-Transfer-Encoding: binary
Hello

--7a9cbec02ceef655
Content-Type: application/octet-stream
Content-Transfer-Encoding: binary
<hex>
3082015206092A86
4886F70D010702A08201433082013F02
010131093007060520E03021A300B06
092A868486F70D010701318201203082
011C020101307C3070310B3009060355
04061302555331133011060355040813
0A4361C69666F7626E691111300F06
03550407130853616E204A6F7365310E
300C060355040A130573697069743129
3027060355040B132053697069742054
657374204365727696696361746520
417574686F726974902080195007102
3301133007060520E03021A300D0609
2A868486F70D0101010500481808EF4
66F948F0522D2E5978E9D95AAE9F2FE
15A06659716292E8DA2AA8D8350A68CE
Test Message 1.12

Test # 1.12: Unusual Reason Phrase
Message Details : unreason

SIP/2.0 200 = 2**3 * 5**2
D0BD0BE20D181D182
D0BE20D0B4D0B5D0B2D18FD0BDD0BED181D182D0BE20D0B4
D0B5D0BD2D18FD182D18C202D20D0BF180D0BED181D182D0
BED0B5</hex>

Via: SIP/2.0/UDP 192.0.2.198;branch=z9hG4bK1324923
Call-ID: unreason.1234ksdfak3j2erwedfsASdf
CSeq: 35 INVITE
From: sip:user@example.com;tag=11141343
To: sip:user@example.edu;tag=2229
Content-Length: 154
Content-Type: application/sdp
Contact: <sip:user@host198.example.com>

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.198
s=-
c=IN IP4 192.0.2.198
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 1.13

Test # 1.13: No Reason Phrase
Message Details : noreason

SIP/2.0 100
Via: SIP/2.0/UDP 192.0.2.105;branch=z9hG4bK2398nda0e
Call-ID: noreason.asndj203insdf99223ndf
CSeq: 35 INVITE
From: <sip:user@example.com>;tag=39ansfi3
To: <sip:user@example.edu>;tag=902jndnke3
Content-Length: 0
Contact: <sip:user@host105.example.com>

Test Message 2.1

Test # 2.1: Extraneous Header Field Separators
Message Details : badinv01
INVITE sip:user@example.com SIP/2.0
To: sip:j.user@example.com
From: sip:caller@example.net;tag=134161461246
Max-Forwards: 7
Call-ID: badinv01.0ha0isndaksdjasdfs3234nas
CSeq: 8 INVITE
Via: SIP/2.0/UDP 192.0.2.15;;,;
Contact: "Joe" <sip:joe@example.org>;;;
Content-Length: 152
Content-Type: application/sdp

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.15
s=-
c=IN IP4 192.0.2.15
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.2
Test # 2.2: Content Length Larger than the Message
Message Details : clerr

INVITE sip:user@example.com SIP/2.0
Max-Forwards: 80
To: sip:j.user@example.com
From: sip:caller@example.net;tag=93942939o2
Contact: <sip:caller@hungry.example.net>
Call-ID: clerr.0ha0isndaksdjweiafasdk3
CSeq: 8 INVITE
Via: SIP/2.0/UDP host5.example.com;branch=z9hG4bK-39234-23523
Content-Type: application/sdp
Content-Length: 9999

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.15
s=-
c=IN IP4 192.0.2.15
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.3
Test # 2.3: Negative Content Length
Message Details : ncl

INVITE sip:user@example.com SIP/2.0
Max-Forwards: 254
To: sip:j.user@example.com
From: sip:caller@example.net;tag=32394234

Voice Over IP Consortium
Call-ID: ncl.0ha0isndaksdj2193423r542w35
CSeq: 0 INVITE
Via: SIP/2.0/UDP 192.0.2.53;branch=z9hG4bKkdjuw
Contact: <sip:caller@example53.example.net>
Content-Type: application/sdp
Content-Length: -999

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.53
s=-
c=IN IP4 192.0.2.53
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.4
Test # 2.4: Request Scalar Fields with Overlarge Values
Message Details : scalar02

REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/TCP host129.example.com;branch=z9hG4bK342sdfoi3
To: <sip:user@example.com>
From: <sip:user@example.com>;tag=239232jh3
CSeq: 36893488147419103232 REGISTER
Call-ID: scalar02.23o0pd9vanlq3wnrlnewofjas9ui32
Max-Forwards: 300
Expires: 1<repeat count=100>0</repeat>
Contact: <sip:user@host129.example.com>
;expires=280297596632815
Content-Length: 0

Test Message 2.5
Test # 2.5: Response Scalar Fields with Overlarge Values
Message Details : scalarlg

SIP/2.0 503 Service Unavailable
<allOneLine>
Via: SIP/2.0/TCP host129.example.com
;branch=z9hG4bKzzxdiwo34sw
;received=192.0.2.129
<br>
To: <sip:user@example.com>
From: <sip:other@example.net>;tag=2easdjfejw
CSeq: 9292394834772304023312 OPTIONS
Call-ID: scalarlg.noase0of0234hn2qsoaf0232aewf2394r
Retry-After: 949302838503028349304023988
Warning: 1812 overtured "In Progress"
Content-Length: 0

Test Message 2.6
Test # 2.6: Unterminated Quoted String
To: "Mr J. User <sip:j.user@example.com> <sip:realj@example.net>

Message Details: quotbal

INVITE sip:user@example.com SIP/2.0
To: "Mr. J. User <sip:j.user@example.com>
From: sip:caller@example.net;tag=93334
Max-Forwards: 10
Call-ID: quotbal.aksdj
Contact: <sip:caller@host59.example.net>
CSeq: 8 INVITE
Via: SIP/2.0/UDP 192.0.2.59:5050;branch=z9hG4bKkdjuw39234
Content-Type: application/sdp
Content-Length: 152

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.15
s=-
c=IN IP4 192.0.2.15
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.7
Test # 2.7: "<>" Enclosing the URI
Message Details: ltgtruri

INVITE <sip:user@example.com> SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=39291
Max-Forwards: 23
Call-ID: ltgtruri.1@192.0.2.5
CSeq: 1 INVITE
Via: SIP/2.0/UDP 192.0.2.5
Contact: <sip:caller@host5.example.net>
Content-Type: application/sdp
Content-Length: 159

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.15
s=-
c=IN IP4 192.0.2.15
t=3149328700 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.8
Test # 2.8: Malformed SIP Request URI
Message Details: lwsruri

INVITE sip:user@example.com; lr SIP/2.0
Test Message 2.9
Test # 2.9: Multiple SP Separating Request Line Elements
Message Details : lwsstart

INVITE sip:user@example.com SIP/2.0
Max-Forwards: 8
To: sip:user@example.com
From: sip:caller@example.net;tag=8814
Call-ID: lwsstart.dfknq234oi243099adsfnawe3@example.com
CSeq: 1893884 INVITE
Via: SIP/2.0/UDP host1.example.com;branch=z9hG4bK299342093
Contact: <sip:caller@host1.example.net>
Content-Type: application/sdp
Content-Length: 150

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.1
s=-
c=IN IP4 192.0.2.1
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 2.10
Test # 2.10: SP Characters at the End of the Request Line
Message Details : trws

OPTIONS sip:remote-target@example.com SIP/2.0
Via: SIP/2.0/TCP host1.example.com;branch=z9hG4bK299342093
To: <sip:remote-target@example.com>
From: <sip:local-resource@example.com>;tag=329429089
Call-ID: trws.oicu3495823rneffasdrh2345r
Accept: application/sdp
CSeq: 238923 OPTIONS
Max-Forwards: 70
Test Message 2.11
Test # 2.11: Escaped Headers (%) in the Request URI
Message Details : escruri

INVITE sip:user@example.com?Route=%3Csip:example.com%3E SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=341518
Max-Forwards: 7
Contact: <sip:caller@host39923.example.net>
Call-ID: escruri.23940-asdfhj-aje3br-234q098w-fawerh2q-h4n5
CSeq: 149209342 INVITE
Via: SIP/2.0/UDP host-of-the-hour.example.com;branch=z9hG4bKkdjuw
Content-Type: application/sdp
Content-Length: 150

Test Message 2.12
Test # 2.12: Invalid Time Zone
Message Details : baddate

INVITE sip:user@example.com SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=2234923
Max-Forwards: 70
Call-ID: baddate.239423mnsadf3j23lj42--sedfnm234
CSeq: 1392934 INVITE
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKkdjuw
Date: Fri, 01 Jan 2010 16:00:00 EST
Contact: <sip:caller@host5.example.net>
Content-Type: application/sdp
Content-Length: 150

Test Message 2.13
Test # 2.13: Name-Addr Format URI without “<>”
Message Details : regbadct
REGISTER sip:example.com SIP/2.0
To: sip:user@example.com
From: sip:user@example.com;tag=998332
Max-Forwards: 70
Call-ID: regbadct.k345asrl3fdbv@10.0.0.1
CSeq: 1 REGISTER
Via: SIP/2.0/UDP 135.180.130.133:5060;branch=z9hG4bKkdjuw
Contact: sip:user@example.com?Route=%3Csip:sip.example.com%3E
l: 0

Test Message 2.14
Test # 2.14: Spaces Within Addr-Spec
Message Details : badaspec

OPTIONS sip:user@example.org SIP/2.0
Via: SIP/2.0/UDP host4.example.com:5060;branch=z9hG4bKkdju43234
Max-Forwards: 70
From: "Bell, Alexander" <sip:a.g.bell@example.com>;tag=433423
To: "Watson, Thomas" <sip:t.watson@example.org>
Call-ID: badaspec.sdf0234n2nds0a099u23h3hnnw009cdkne3
Accept: application/sdp
CSeq: 3923239 OPTIONS
l: 0

Test Message 2.15
Test # 2.15: Non-token Characters in the Display Name
Message Details: badnn

OPTIONS sip:t.watson@example.org SIP/2.0
Via: SIP/2.0/UDP c.example.com:5060;branch=z9hG4bKkdjuw
Max-Forwards: 70
From: Bell, Alexander <sip:a.g.bell@example.com>;tag=43
To: Watson, Thomas <sip:t.watson@example.org>
Call-ID: badn.31415@c.example.com
Accept: application/sdp
CSeq: 3923239 OPTIONS
l: 0

Test Message 2.16
Test # 2.16: Unknown Protocol Version
Message Details: badvers

OPTIONS sip:t.watson@example.org SIP/7.0
Via: SIP/7.0/UDP c.example.com;branch=z9hG4bKkdjuw
Max-Forwards: 70
From: A. Bell <sip:a.g.bell@example.com>;tag=qweoiqpe
To: T. Watson <sip:t.watson@example.org>
Call-ID: badvers.31417@c.example.com
CSeq: 1 OPTIONS
l: 0
Test Message 2.17  
Test # 2.17: Request Line and CSeq Method Mismatch  
Message Details : mismatch01

OPTIONS sip: user@example.com SIP/2.0  
To: sip: j.user@example.com  
From: sip: caller@example.net;tag=34525  
Max-Forwards: 6  
Call-ID: mismatch01.dj0234sxdfl3  
CSeq: 8 INVITE  
Via: SIP/2.0/UDP host.example.com;branch=z9hG4bKkdjuw  
l: 0

Test Message 2.18  
Test # 2.18: Unknown Method with CSeq Method Mismatch  
Message Details : mismatch02

NEWMETHOD sip: user@example.com SIP/2.0  
To: sip: j.user@example.com  
From: sip: caller@example.net;tag=34525  
Max-Forwards: 6  
Call-ID: mismatch02.dj0234sxdfl3  
CSeq: 8 INVITE  
Contact: <sip: caller@host.example.net>  
Via: SIP/2.0/UDP host.example.net;branch=z9hG4bKkdjuw  
Content-Type: application/sdp  
l: 138

v=0  
o=mhandley 29739 7272939 IN IP4 192.0.2.1  
c=IN IP4 192.0.2.1  
m=audio 49217 RTP/AVP 0 12  
m=video 3227 RTP/AVP 31  
a=rtpmap:31 LPC

Test Message 2.19  
Test # 2.19: Overlarge Response Code  
Message Details : bigcode

SIP/2.0 4294967301 better not break the receiver  
Via: SIP/2.0/UDP 192.0.2.105;branch=z9hG4bK2398nda0e  
Call-ID: bigcode.asdof3uj203asdnf3429uasdhfas3ehjasdfas9i  
CSeq: 353494 INVITE  
From: <sip: user@example.com>;tag=39ansfi3  
To: <sip: user@example.edu>;tag=902jndnke3  
Content-Length: 0  
Contact: <sip: user@host105.example.com>

Test Message 3.1  
Test # 3.1: Missing Transaction Identifiers  
Message Details : badbranch

OPTIONS sip: user@example.com SIP/2.0  
To: sip: user@example.com  
From: sip: caller@example.org;tag=33242
Max-Forwards: 3
Via: SIP/2.0/UDP 192.0.2.1;branch=z9hG4bK
Accept: application/sdp
Call-ID: badbranch.sadonfo23i420jv0as0derf3j3n
CSeq: 8 OPTIONS
l: 0

Test Message 4.1
Test # 4.1: Missing Required Header Fields
Message Details : insuf

INVITE sip:user@example.com SIP/2.0
CSeq: 193942 INVITE
Via: SIP/2.0/UDP 192.0.2.95;branch=z9hG4bKkdj.insuf
Content-Type: application/sdp
l: 152

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.95
s=-
c=IN IP4 192.0.2.95
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 4.2
Test # 4.2: Request URI with Unknown Scheme
Message Details : unkscm

OPTIONS nobodyKnowsThisScheme:totallyopaquecontent SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=384
Max-Forwards: 3
Call-ID: unkscm.nasdfasser0q239nwsdfasdkl34
CSeq: 3923423 OPTIONS
Via: SIP/2.0/TCP host9.example.com;branch=z9hG4bKkdjuw39234
Content-Length: 0

Test Message 4.3
Test # 4.3: Request URI with Known But Atypical Scheme
Message Details : novelsc

OPTIONS soap.beep://192.0.2.103:3002 SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=384
Max-Forwards: 3
Call-ID: novelsc.asdfasser0q239nwsdfasdkl34
CSeq: 3923423 OPTIONS
Via: SIP/2.0/TCP host9.example.com;branch=z9hG4bKkdjuw39234
Content-Length: 0

Test Message 4.4
Test # 4.4: Unknown URI in Header Scheme  
Message Details : unksm2

REGISTER sip:example.com SIP/2.0  
To: isbn:2983792873  
From: <http://www.example.com>;tag=3234233  
Call-ID: unksm2.daksdj@hyphenated-host.example.com  
CSeq: 234902 REGISTER  
Max-Forwards: 70  
Via: SIP/2.0/UDP 192.0.2.21:5060;branch=z9hG4bKkdjuw  
Contact: <name:John_Smith>  
l: 0

Test Message 4.5
Test # 4.5: Unsupported Requires and Proxy-Requires  
Message Details : bext01

OPTIONS sip:user@example.com SIP/2.0  
To: sip:j_user@example.com  
From: sip:caller@example.net;tag=242etr  
Max-Forwards: 6  
Call-ID: bext01.0ha0isndaksdj  
Require: nothingSupportsThis, nothingSupportsThisEither  
Proxy-Require: noProxiesSupportThis, norDoAnyProxiesSupportThis  
CSeq: 8 OPTIONS  
Via: SIP/2.0/TLS fold-and-staple.example.com;branch=z9hG4bKkdjuw  
Content-Length: 0

Test Message 4.6
Test # 4.6: Unknown Content Type  
Message Details : invut

INVITE sip:user@example.com SIP/2.0  
Contact: <sip:caller@host5.example.net>  
To: sip:j_user@example.com  
From: sip:caller@example.net;tag=8392034  
Max-Forwards: 70  
Call-ID: invut.0ha0isndaksdjadsfj34n23d  
CSeq: 235448 INVITE  
Via: SIP/2.0/UDP somehost.example.com;branch=z9hG4bKkdjuw  
Content-Type: application/unknownformat  
Content-Length: 40

<audio>  
<pcmu port="443"/>  
</audio>

Test Message 4.7
Test # 4.7: Unknown Authorization Scheme  
Message Details : regaut01

REGISTER sip:example.com SIP/2.0  
To: sip:j_user@example.com  
From: sip:j_user@example.com;tag=87321hj23128
Test Message 4.8

Test # 4.8: Multiple Values in Single Value Fields
Message Details : multi01

INVITE sip:user@company.com SIP/2.0
Contact: <sip:caller@host25.example.net>
Via: SIP/2.0/UDP 192.0.2.25;branch=z9hG4bKkdjuw
Max-Forwards: 70
CSeq: 5 INVITE
Call-ID: multi01.98asdh@192.0.2.1
CSeq: 59 INVITE
Content-Type: application/sdp
l: 154
Contact: <sip:caller@host36.example.net>
Max-Forwards: 5

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.25
s=-
c=IN IP4 192.0.2.25
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 4.9

Test # 4.9: Multiple Content Length Values
Message Details : mcl01

OPTIONS sip:user@example.com SIP/2.0
Via: SIP/2.0/UDP host5.example.net;branch=z9hG4bK293423
To: sip:user@example.com
From: sip:other@example.net;tag=3923942
Call-ID: mcl01.fhn2323orihawfdoa3o4r52o3irsdf
CSeq: 15932 OPTIONS
Content-Length: 13
Max-Forwards: 60
Content-Type: text/plain

Test Message 4.10

Test # 4.10: Response with Broadcast Via Header Field Value
SIP/2.0 200 OK
Via: SIP/2.0/UDP 192.0.2.198;branch=z9hG4bK1324923
Via: SIP/2.0/UDP 255.255.255.255;branch=z9hG4bKlsaber23
Call-ID: bcast.0384840201234ksdfak3j2erwedfsASdf
CSeq: 35 INVITE
From: sip:user@example.com;tag=11141343
To: sip:user@example.edu;tag=2229
Content-Length: 154
Content-Type: application/sdp
Contact: <sip:user@host28.example.com>

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.198
s=-
c=IN IP4 192.0.2.198
t=0 0
m=audio 49217 RTP/AVP 0 12
m=video 3227 RTP/AVP 31
a=rtpmap:31 LPC

Test Message 4.11
Test # 4.11: Max-Forward of Zero
Message Details : zeromf

OPTIONS sip:user@example.com SIP/2.0
To: sip:user@example.com
From: sip:caller@example.net;tag=3ghsd41
Call-ID: zeromf.jfasdlfnm2o2l43r5u0asdfas
CSeq: 39234321 OPTIONS
Via: SIP/2.0/UDP host1.example.com;branch=z9hG4bKkdjuw2349i
Max-Forwards: 0
Content-Length: 0

Test Message 4.12
Test # 4.12: REGISTER with a Contact Header Parameter
Message Details : cparam01

REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP saturn.example.com:5060;branch=z9hG4bKkdjuw
Max-Forwards: 70
From: sip:watson@example.com;tag=DkfVgjktMwaerKKpe
To: sip:watson@example.com
Call-ID: cparam01.70710@saturn.example.com
CSeq: 2 REGISTER
Contact: sip: +19725552222@gw1.example.net;unknownparam
l: 0

Test Message 4.13
Test # 4.13: REGISTER with a URL Parameter
Message Details : cparam02

REGISTER sip:example.com SIP/2.0
Via: SIP/2.0/UDP saturn.example.com:5060;branch=z9hG4bKkdjuw
Max-Forwards: 70
From: sip:watson@example.com;tag=838293
To: sip:watson@example.com
Call-ID: cparam02.70710@saturn.example.com
CSeq: 3 REGISTER
Contact: <sip:+19725552222@gw1.example.net;unknownparam>
l: 0

Test Message 4.14
Test # 4.14: REGISTER with a URL Escaped Header
Message Details : regescrt

   REGISTER sip:example.com SIP/2.0
   To: sip:user@example.com
   From: sip:user@example.com;tag=8
   Max-Forwards: 70
   Call-ID: regescrt.k345asrl3fdiv@192.0.2.1
   CSeq: 14398234 REGISTER
   Via: SIP/2.0/UDP host5.example.com;branch=z9hG4bKkdjuw
   M: <sip:user@example.com?Route=%3Csip:sip.example.com%3E>
l: 0

Test Message 4.15
Test # 4.15: Unacceptable Accept Offering
Message Details : sdp01

   INVITE sip:user@example.com SIP/2.0
   To: sip:user@example.com
   Contact: <sip:caller@host15.example.net>
   From: sip:caller@example.net;tag=234
   Max-Forwards: 5
   Call-ID: sdp01.ndaksdj9342dasdd
   Accept: text/nobodyKnowsThis
   CSeq: 8 INVITE
   Via: SIP/2.0/UDP 192.0.2.15;branch=z9hG4bKkdjuw
   Content-Length: 150
   Content-Type: application/sdp

   v=0
   o=mhandley 29739 7272939 IN IP4 192.0.2.5
   s=-
   c=IN IP4 192.0.2.5
   t=0 0
   m=audio 49217 RTP/AVP 0 12
   m=video 3227 RTP/AVP 31
   a=rtpmap:31 LPC

Test Message 5.1
Test # 5.1: INVITE with RFC 2543 Syntax
Message Details : inv2543

   INVITE sip:UserB@example.com SIP/2.0
   Via: SIP/2.0/UDP iftgw.example.com

VoIP Consortium
75
SIP Torture Test Suite
From: <sip:+13035551111@ift.client.example.net;user=phone>
Record-Route: <sip:UserB@example.com; maddr=ss1.example.com; user=phone>
To: sip:+16505552222@ss1.example.net; user=phone
Call-ID: inv2543.1717@ift.client.example.com
CSeq: 56 INVITE
Content-Type: application/sdp

v=0
o=mhandley 29739 7272939 IN IP4 192.0.2.5
s=-
c=IN IP4 192.0.2.5
t=0 0
m=audio 49217 RTP/AVP 0