### Overview

This guide explains the interface specification used for Customization of SIP messages on Cisco Unified CM- Session Management (SME). It includes details on Lua environment available on Cisco Unified CM-SME and APIs to support SIP Transparency and Normalization functionality.

This chapter describes the following topics:

- Introduction
- Scripting Environment
- Message Handlers

### Introduction

This chapter describes Lua Environment, the interface used for customizing Session Management (SM) behavior for a particular deployment. Lua is a non-proprietary, lightweight scripting language. It is assumed that the reader of this guide has basic familiarity with the Lua scripting language.

Cisco Unified CM provides a set of features called SIP Normalization and Transparency to customize the SIP messages:

- **Normalization**—It is the process of transforming inbound and outbound messages.
  - For inbound messages, normalization happens after having received the message from the network. The inbound message normalization is used to make the message more palatable to Cisco Unified CM. For example, Cisco Unified CM supports Diversion header for carrying redirecting number information. Some SIP devices connected to Cisco Unified CM use the History-Info header for this purpose. Inbound normalization transforms the History-Info headers into Diversion headers so that Cisco Unified CM recognizes the redirecting information.
  - For outbound messages, normalization happens just prior to sending the message to the network. Thus outbound message normalization is used to make the message more palatable to an external SIP device (e.g. another SIP capable PBX). For example, outbound normalization can be used to transform Diversion headers into History-Info headers so that the external SIP device will recognize the redirecting information.

- **Transparency**—It allows SIP information to be passed through from one call leg to another even though Cisco Unified CM is a Back to Back User Agent (B2BUA).
The Normalization and Transparency features is exposed by a script which is associated with a SIP trunk or a SIP line. The scripts manifest themselves as a set of message handlers that operate on inbound and outbound SIP messages. For normalization, the script manipulates almost every aspect of a SIP message including:

- The request URI
- The response code and phrase
- SIP headers
- SIP parameters
- Content bodies
- SDP

For transparency, the script passes through almost any information in a SIP message including:

- SIP headers
- SIP parameters
- Content bodies

This guide describes the scripting environment and APIs used to manipulate and pass through SIP message information.

For more information on script management, see the *Cisco Unified Communications Manager Administration Guide*.

**Scripting Environment**

The interface for customizing Cisco Unified CM SIP Trunk behavior is provided by a scripting language called Lua. Lua is an open source, lightweight scripting language.

The Lua Environment available for Cisco Unified CM (SM) is a restricted sub-set of Lua. In addition to the basic capabilities provided by Lua, like:

- lexical conventions
- values and types
- variables
- statements
- expressions, etc

Lua also provides some basic libraries, like:

- base
- co-routine
- modules
- string manipulation
- table manipulation
- mathematical functions
- OS facilities
- IO facilities and
- debug capabilities
However, the Cisco SIP Lua Environment available on SM only supports the string library in its entirety and a subset of the base library. The other libraries are not supported.

For the base library, the following is supported:

- ipairs
- pairs
- next
- unpack
- error
- type
- tostring

The Cisco SIP Lua Environment provides a global environment for the scripts to use, it does not expose the default Lua global environment (i.e. _G) to the scripts.

The Lua script provides a set of call back functions called **message handlers** to manipulate SIP messages in the context of SM environment. The name of the message handler indicates the handler that is invoked for a particular SIP message. For example, the script's “inbound_INVITE” message handler is invoked when an inbound INVITE is received by Cisco Unified CM. The message handlers receive a single parameter called **msg** representing a SIP Message. The Lua script uses the APIs defined in Cisco SIP Message library to access and manipulate the **msg** parameter.

The following part of the guide describes the details on the **message handler** construct. The next section has details on the Cisco SIP Message library API's.

Let's take a look at an example script that simply removes the "Cisco-Guid" header in an outbound INVITE. The script is shown with line numbers on left for ease of describing the script.

**Simple Script: M.lua**

1. M = {}
2. function M.inbound_INVITE(msg)
3.     msg:convertHIToDiversion()
4. end
5. function M.outbound_INVITE(msg)
6.     msg:removeHeader("Cisco-Guid")
7. end
8. return M

There are three important parts to the above script:

1. Module Initialization—The first line of the script creates a Lua table called 'M' and initializes it to be empty. This table contains set of callback functions provided by this script. The variable **M** is a Lua table and is also the name of the module.

2. Message Handler Definitions—Lines 2-4 defines an inbound INVITE **message handler**. This callback function is executed when an inbound INVITE is received out on the SIP trunk or SIP line associated with this script. In this example, the message handler invokes an API to convert History-Info headers into Diversion headers.

   Lines 5-6 defines an outbound INVITE **message handler**. This callback function is executed when an outbound INVITE is sent out on the SIP trunk or SIP line associated with this script. In this example, the message handler invokes an API to remove the "Cisco-Guid" header.

   The script can define multiple message handlers. The name of the message handler dictates which message handler is invoked (if any) for a given SIP message.
3. Returning the Module—The last line returns the name of the module. This line is absolutely required. This is how the Cisco SIP Lua Environment finds the message handlers associated with the script.

Message Handlers

The Lua script provides a set of call back functions called message handlers to manipulate SIP messages in the context of SM environment. The name of the message handler indicates which handler is invoked for a particular SIP message.

Naming

The naming rules for message handlers dictate which message handler is invoked for a given SIP message. Various SIP messages by specification, are split into requests and responses.

- For requests—the message handler is named according to the message direction and the SIP request method name. The method name is the one in the 'request line' of SIP message.

\(<\text{direction}>\_<\text{method}>\)

Example

- inbound_INVITE
- outbound_UPDATE

- For responses—the message handler is named according to the message direction, the response code, and the SIP method. For responses, the method name is obtained from the CSeq header of SIP message.

\(<\text{direction}>\_<\text{response code}>\_<\text{method}>\)

Example

- inbound_183_INVITE
- inbound_200_INVITE
- outbound_200_UPDATE

Use Case

Consider the case where a Cisco Unified CM-SME is connected to PBX-A and PBX-B via two trunks. A script that returns module A is attached to trunk connecting to PBX-A. Similarly, a script that returns module B is attached to trunk connecting to PBX-B.
The following handlers are executed for an INVITE dialog.

Wild cards

The message handler names also support wild carding. The wild card support is dependent on whether the message is request or response SIP message.

Request messages

A wildcard ANY can be used in place of <method>. The <direction> does not support wild card.

Refer to Table 1-1 on page 1-5 for valid request message handler names.

Table 1-1  Valid Request Message Handler Names

<table>
<thead>
<tr>
<th>Message Handler</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>M.inbound_INVITE</td>
<td>This message handler is invoked for all inbound INVITE messages including initial INVITEs and reINVITEs.</td>
</tr>
<tr>
<td>M.inbound_ANY</td>
<td>This message handler is invoked for all inbound requests.</td>
</tr>
<tr>
<td>M.outbound_ANY</td>
<td>This message handler is invoked for all outbound requests.</td>
</tr>
</tbody>
</table>

Response messages

A wildcard ANY can be used in place of <method> and/or <response code>. The <direction> does not support wild card. Additionally, a wildcard character X can be used in <response code>.

Refer to Table 1-2 on page 1-6 for valid response message handler names.
Chapter 1  Overview

Message Handlers

Rules to Pick Message Handler

Cisco Unified CM uses the following rules to find the message handler for a given message:

1. Message handlers are case-sensitive.
2. The direction is either inbound or outbound.
3. The direction is always written as lowercase.
4. The message direction is relative to SM.

Note The message direction is completely separate from dialog direction in SIP.

Example
inbound_INVITE is valid handler name; whereas InBound_INVITE is NOT a valid handler name

5. The method name obtained from SIP message is converted to uppercase for the purpose of finding the appropriate message handler in the script.

6. CUCM-SME uses a longest-match criterion to find the correct message handler.

Example
Assume a script has two message handlers; inbound_ANY_ANY and inbound_183_INVITE. When a 183 response is received by the Cisco Unified CM, the inbound_183_INVITE handler will be executed since it is the most explicit match.

Note Although inbound or outbound is supported with ANY (response code) and ANY (method), we do not currently support the ANY (method) wildcard with specific response codes.

In other words, the following message handlers are valid:

inbound_ANY_ANY
outbound_ANY_ANY

But these are invalid:
inbound_401_ANY
outbound_200_ANY
etc.