SIP Message Manipulation

You can configure the Cisco Unified Border Element (SP Edition) to selectively examine and manipulate incoming SIP messages on an adjacency.

Cisco Unified Border Element (SP Edition) was formerly known as Integrated Session Border Controller and may be commonly referred to in this document as the session border controller (SBC).

For a complete description of the commands used in this chapter, refer to Cisco Unified Border Element (SP Edition) Command Reference: Unified Model at:


For information about all Cisco IOS commands, use the Command Lookup Tool at http://tools.cisco.com/Support/CLILookup or a Cisco IOS master commands list.

Feature History for SIP Header Manipulation on Cisco Unified Border Element (SP Edition)

<table>
<thead>
<tr>
<th>Release</th>
<th>Modification</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cisco IOS XE Release 2.4</td>
<td>The SIP Header Profile, SIP Method Profile, Parameter Profile, Response Code Mapping, SIP Header Manipulation, and Provisional Response filtering features were introduced on Cisco IOS XR along with support for the unified model.</td>
</tr>
</tbody>
</table>
| Cisco IOS XE Release 2.5 | The following features were introduced on the Cisco ASR 1000 Series Routers:  
  • Ability to Insert Firewall Parameter in SIP Contact Header.  
  • Enhanced SIP header manipulation functionality on the Cisco ASR 1000 Series Routers.  
  • P-KT-UE-IP header (type of private header) support as part of SIP header manipulation functionality. |
| Cisco IOS XE Release 2.6 | The following SIP header manipulation functions were enabled with new CLIs on the Cisco ASR 1000 Series Routers:  
  • Parse User Name Parameters  
  • Suppress Expires Header  
  • Configuring Customer P-Asserted-Identity |
| Cisco IOS XE Release 3.1S | The following features were added on the Cisco ASR 1000 Series Routers:  
  • SIP Destination ID  
  • SIP Source ID |
This chapter contains the following sections:

- SIP Message Editing Using Profiles, page 23-3
- SIP Message Editing Using Editors, page 23-64
- SDP Editing Using Script-Based Editors, page 23-84

Cisco IOS XE Release 3.2S  SBC supports call-policy routing of calls using the hostname in the Request URI. The calls are now routed even in the absence of username in the Request URI.

The Event Header in Publish Method feature was added on the Cisco ASR 1000 Series Routers.

Cisco IOS XE Release 3.3S  The SIP Message Editing feature was added.

Cisco IOS XE Release 3.4S  The SDP Editing Using Script-Based Editors feature was added.
SIP Message Editing Using Profiles

This section contains the following information on SIP profiles:

- Information About SIP Profiles, page 23-3
- Method Profiles, page 23-4
- Response Code Mapping, page 23-12
- Header Profiles, page 23-16
- Provisional Response Filtering, page 23-33
- Parameter Profiles, page 23-36
- Ability to Insert Firewall Parameter in the SIP Contact Header, page 23-42
- Configuration Examples for SIP Profiles, page 23-46

Note

From Release 3.3S, the concept of editors has been introduced. An editor is the enhanced version of its corresponding profile. From SIP Message Editing Using Editors? section on page 23-64, all occurrences of profile have been replaced by editor. For example, a method profile is called a method editor.

Information About SIP Profiles

Cisco Unified Border Element (SP Edition) can manipulate the following SIP profiles:

- Method profiles
- Header profiles
- Parameter profiles

Method profiles allow the association of header profiles and parameter profiles to method elements contained in the method profile. You can use actions with method profiles to allow the whitelist to contain blacklisted headers and the blacklist to contain whitelisted headers as well as to reject non-vital methods. This allows any profile to contain mixed actions per-profile.

Header profiles allow complex header manipulation to occur, over and above the existing whitelist and blacklist functionality using actions based on conditional expressions.

Header profiles additionally allow the association of parameter profiles in header elements contained in the profile.

You can use variables to store header content; you can then optionally reconstruct the headers using previously stored variables. You can also match headers based on regular expression matching. You can use conditional matching to match against adjacency settings, transport addresses, and a number of boolean match criteria. You can also use header profiles to reference and make limited modifications to the Request Line.

A header profile can conditionally match any part of a header, but can only replace the entire header. SIP parameter profiles extend this capability to allow changes to be made to individual SIP Request Uniform Resource Identifier (URI) parameters associated with a header.

Parameter profiles allow the removal, replacement, or addition of specific URI parameters within certain vital headers.
You can also associate parameter profiles with methods in method profiles for the purpose of request-line processing per method only.

You can configure multiple store rules, request-lines, and header entries, each with unique actions and/or conditions under which the action is applied. Figure 23-1 shows the hierarchical association of adjacency, method profiles, header profiles, and parameter profiles. The dotted line shows the deprecated method for parameter profile association to method profiles.

**Figure 23-1  SIP Profiles**

Method Profiles

SIP methods can be blacklisted and whitelisted dynamically at run-time during receipt of a message (ingress) and at transmission of a message (egress).

A configured method profile allows two types of method profiles for non-vital requests. These can be blacklist (drop) or whitelist (pass). The whitelist action is considered to be the default type for a method if ‘blacklist’ is not present in the command line.

The method profile will contain a list of methods which are either passed on (whitelist) or dropped (blacklist). A single profile can then be associated with each of the inbound or outbound call sides.

Method profiles can be associated with pre-defined header profiles. In addition, pre-defined parameter profiles can be associated with the Request-line per method.

Method profiles are not allowed to blacklist or whitelist vital methods; however, header profiles and parameter profiles can be associated with vital methods.
Status code mapping can be associated with any method type declared in a method profile such that any response identified with this method can be changed. For example, a 503 response to an INVITE could potentially be changed to a 500 response if appropriate mapping is declared against the INVITE method.

This section contains the following topics:

- Restrictions for Configuring Method Profiles, page 23-5
- Information About Method Profiles, page 23-5
- Configuring Method Profiles, page 23-7
- Unconfiguring Method Profiles, page 23-9
- Applying Method Profiles, page 23-11

Restrictions for Configuring Method Profiles

Review the following restrictions for method profiles:

- Any given profile must be exclusively a whitelist or a blacklist.
- Two profiles are applied to process any given SIP message: one inbound and, if permitted through that, one outbound.
- Profiles check only SIP methods in the Request Uniform Resource Identifier (URI).
- SIP requests that are blacklisted and non-essential are rejected as a result of a method profile’s rules. SIP responses are always forwarded.
- Any method unknown to Cisco Unified Border Element (SP Edition) which is forwarded as a result of a profile’s rules does not affect creating or deleting a SIP dialog.
- Methods that are essential to the operation of Cisco Unified Border Element (SP Edition) cannot be blacklisted and are implicitly added to any whitelist.
- Profiles cannot be deleted while they are in active use by at least one adjacency.
- In case of non-Information Management System (IMS) preset, there is a default method profile (sip method-profile default). If configured, the default method profile is attached to the adjacencies for which no explicit user-defined method profiles are configured for both inbound and outbound. The sip method profile default is an empty white-list by itself.

Information About Method Profiles

After you configure a profile, you can assign it for a default application. Any SIP adjacency can apply it to signaling for that adjacency.

Profiles are an optional part of the configuration—they do not have to be specified for Cisco Unified Border Element (SP Edition) to operate correctly. The default behavior is that requests with one of the essential methods are processed, and all other requests are rejected.

You can add or remove methods from profiles at any time. Each method can optionally be assigned one of three actions with the action command:

- Either pass or reject the method.
- Use the as-profile action to select the default profile blacklist or whitelist.

Profiles cannot be deleted while at least one adjacency is using them. You can see which adjacencies are using a profile by entering the following show commands:
show sbc abc-name sbe sip method-profile [profile-name]
or
show sbc abc-name sbe sip essential-methods
The following methods are part of the essential method set:

- ACK
- BYE
- CANCEL
- INVITE
- NOTIFY
- PRACK
- REFER
- REGISTER
- SUBSCRIBE

To modify parameters in the request-line, associate a parameter profile with a method profile.

Cisco IOS XE Release 2.4 and later contains the following functionalities:

- Predefined header profiles can be associated with outgoing method profiles.
- Predefined parameter profiles can be associated with the request-line per method.

**Note**

Header profiles and parameter profiles can be associated with essential methods even though method profiles are not allowed to blacklist/whitelist essential methods.

- Response code mapping can be associated with any method type declared in a method profile so that any response identified with the method can be changed. For example, a 503 response to an INVITE could potentially be changed to a 500 response if appropriate mapping is declared against the INVITE method.

### Configuring Method Profiles

This procedure shows how to configure method profiles.

**SUMMARY STEPS**

1. configure
2. sbc  *sbc-name*
3. sbe
4. sip method-profile  *profile-name*
5. description  *description*
6. blacklist
7. pass-body
8. method  *name*
9. action \{ as-profile \ |
   pass \ |
   reject \}
10. end
11. show sbc  *sbc-name* sbe sip method-profile [ *profile-name* ]
12. show sbc  *sbc-name* sbe sip essential-methods
# Detailed Steps

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1 configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 2 sbc sbc-name</td>
<td>Enters the submode for configuring the method profile. Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config)# sbc mysbc</td>
<td></td>
</tr>
<tr>
<td>Step 3 sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td>Step 4 sip method-profile profile-name</td>
<td>Configures a method profile and enters SIP method profile configuration mode. If you enter the profile-name default, the default profile is configured. This profile is used for all adjacencies that do not have a specific profile configured.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe)# sip method-profile profile1</td>
<td></td>
</tr>
<tr>
<td>Step 5 description description</td>
<td>Adds a description for the specified profile. The no form of this command removes the description. This description is displayed when the show command is used for this profile and is displayed for each profile when displaying a summary of all profiles.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-mth)# description mysbc profile1</td>
<td></td>
</tr>
<tr>
<td>Step 6 blacklist</td>
<td>Configures a profile to be a blacklist. The no form of this command configures the profile to be a whitelist.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-mth)# blacklist</td>
<td></td>
</tr>
<tr>
<td>Step 7 pass-body</td>
<td>Permissions message bodies to be passed through for non-vital methods accepted by this profile. The no form of this command strips the message body out of any non-vital SIP messages matched by this profile. Non-vital method is same as non-essential method.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-mth)# pass-body</td>
<td></td>
</tr>
<tr>
<td>Step 8 method name</td>
<td>Adds a method with the specified name to the profile. Enters the SBE method profile element configuration mode. This field can be 1 to 32 characters (inclusive) in length and is case-insensitive. The no form of this command deletes the method with that name from the profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-mth)# method test</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 23  SIP Message Manipulation


SIP Message Editing Using Profiles

Unconfiguring Method Profiles

The following example shows the proper sequence for unconfiguring a method profile applied to an adjacency. References to the profile must first be removed from all adjacencies. In this example, only one adjacency refers to the profile.

SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. adjacency sip adjacency-name
5. no method-profile inbound profile-name
6. exit
7. no sip method-profile profile name
8. end

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 9 action (as-profile</td>
<td>pass</td>
</tr>
<tr>
<td>Example: Router(config-sbc-sbe-sip-mth-ele)# action as-profile</td>
<td></td>
</tr>
<tr>
<td>Step 10 end</td>
<td>Exits SBE method profile element configuration mode and returns to Privileged EXEC mode.</td>
</tr>
<tr>
<td>Example: Router(config-sbc-sbe-sip-mth-ele)# end</td>
<td></td>
</tr>
<tr>
<td>Step 11 show sbc sbc-name sbe sip method-profile [profile-name]</td>
<td>Displays details for the method profile with the designated name. Use profile-name default to view the default profile. Displays a list of all configured method profiles if no profile-name is specified.</td>
</tr>
<tr>
<td>Example: Router# show sbc mysbc sbe sip-method-profile profile1</td>
<td></td>
</tr>
<tr>
<td>Step 12 show sbc sbc-name sbe sip essential-methods</td>
<td>Displays a list of the essential methods.</td>
</tr>
<tr>
<td>Example: Router# show sbc mysbc sbe sip essential-methods</td>
<td></td>
</tr>
</tbody>
</table>
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><code>configure terminal</code></td>
<td>Enables the global configuration mode.</td>
</tr>
</tbody>
</table>
|      | **Example:**  
|      | Router# configure terminal | |
| 2    | `sbc sbc-name` | Enters the submode for configuring the method profile. Use the `sbc-name` argument to define the name of the service. |
|      | **Example:**  
|      | Router(config)# sbc mySBC | |
| 3    | `sbe` | Enters the mode of an SBE entity within an SBC service. |
|      | **Example:**  
|      | Router(config-sbc)# sbe | |
| 4    | `adjacency sip adjacency-name` | Enters the mode of an SBE SIP adjacency. Use the `adjacency-name` argument to define the name of the service. |
|      | **Example:**  
|      | Router(config-sbc-sbe)# adjacency sip sipadj1 | |
| 5    | `no method-profile inbound profile-name` | Unconfigures profile1 that was used for inbound signaling on adjacency test. |
|      | **Example:**  
|      | Router(config-sbc-sbe-adj-sip)# no method-profile inbound profile1 | |
| 6    | `exit` | Exits SBE SIP adjacency configuration mode and enters SBE configuration mode. |
|      | **Example:**  
|      | Router(config-sbc-sbe-adj-sip)# exit | |
| 7    | `no sip method-profile profile name` | The `no` form of this command deletes the method with that name from the profile. |
|      | **Example:**  
|      | Router(config-sbc-sbe)# no sip method-profile profile1 | |
| 8    | `end` | Exits the SBE mode and returns to Privileged EXEC mode. |
|      | **Example:**  
|      | Router(config-sbc-sbe)# end | |
Applying Method Profiles

This procedure shows how to apply method profiles.

**SUMMARY STEPS**

1. `configure terminal`
2. `sbc `sbc-name``
3. `sbe`
4. `adjacency sip adjacency-name`
5. `method-profile inbound profile-name`
6. `end`
7. `show sbc `sbc-name`` sbe sip method-profile name`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 1</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>configure terminal</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>Enters the mode of an SBC service.</td>
</tr>
<tr>
<td><strong>sbc `sbc-name``</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config)# sbc mySbc</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>sbe</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>Enters the mode of an SBE SIP adjacency.</td>
</tr>
<tr>
<td><strong>adjacency sip `adjacency-name``</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe)# adjacency sip test</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>Sets profile1 to be used for inbound signaling on adjacency test.</td>
</tr>
<tr>
<td><strong>method-profile inbound profile-name</strong></td>
<td></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-adj-sip)# method-profile inbound profile1</td>
<td></td>
</tr>
</tbody>
</table>

**Note** When attaching a method profile to an adjacency, the adjacency must be in the “no attach” state.
Response Code Mapping

Response code mapping provides an ability to manipulate the SIP response codes when the messages traverse the Cisco Unified Border Element (SP Edition). The mapping table is applied to inbound messages received at a SIP adjacency or to responses sent out of a SIP adjacency. The mapping is user-configurable on a per SIP method basis so that each SIP method can be mapped differently. lists the mapping limitations on SIP response code.

<table>
<thead>
<tr>
<th>Response Codes</th>
<th>Mapping</th>
</tr>
</thead>
<tbody>
<tr>
<td>100</td>
<td>No mapping allowed</td>
</tr>
<tr>
<td>1xx</td>
<td>Maps to 1yy (not 100)</td>
</tr>
<tr>
<td>2xx</td>
<td>Maps to 2yy</td>
</tr>
<tr>
<td>3xx</td>
<td>Maps to 3yy</td>
</tr>
<tr>
<td>4xx</td>
<td>Maps to 4yy, 5yy, or 6yy</td>
</tr>
<tr>
<td>5xx</td>
<td>Maps to 4yy, 5yy, or 6yy</td>
</tr>
<tr>
<td>6xx</td>
<td>Maps to 4yy, 5yy, or 6yy</td>
</tr>
</tbody>
</table>

Response code mapping allows you to:

- Map a particular response code to a specific response code. For example, you can map 401 to 400, but not to 300. You can map 102 to 101, but not 100.
- Map a group of response codes (defined using a wildcard) to a specific response code. For example, you can map 40X to 400, or map all of 4XX to 400.
- Specify exceptions to the wildcard. For example, mapping 2XX to 201, and mapping 200 to 200.

You can use the `map-status-code` command to add one of more mappings.

Where configuration causes the response code to be mapped to one that is not defined in RFC 3261, Cisco Unified Border Element (SP Edition) applies the reason phrase "Unrecognized status code."

This section contains the following topics:

- Configuring Response Code Mapping, page 23-13
- Applying Response Code Mapping, page 23-15
Restrictions for Response Code Mapping

The following restrictions apply to Response Code Mapping:

- Response code mapping only covers mapping of SIP response codes. H.323 calls cannot have their response codes mapped.
- Certain messages are processed only by the SIP Transaction Manager; mapping of these messages is not possible. For example, badly formatted messages that cannot be interpreted are responded to directly by the SIP Transaction Manager.
- There is no provision for the mapping of SIP reason phrases. The reason phrase will always match the reason code as defined in RFC 3261. A generic reason phrase is applied when the requested reason code has no corresponding definition in RFC 3261. This phrase is a compile time constant.
- Changing the response code could result in an invalid message (for example, mapping the response code could produce a message with mandatory headers missing). There is no provision to ensure that messages contain headers required by the new response code.
- A maximum of 128 mappings is permitted in each direction per adjacency (128 inbound and 128 outbound mappings).

Configuring Response Code Mapping

This procedure shows how to configure response code mapping.

SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip method-profile profile-name
5. method name
6. map-status-code
7. range statuscoderange value statuscodevalue
8. end
9. show sbc sbc-name sbe sip method-profile [profile-name]
10. show sbc sbc-name sbe sip essential-methods
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
</tbody>
</table>

**Example:**

```
Router# configure terminal
```

| **Step 2** sbc sbc-name | Enters the submode for configuring the method profile. Use the `sbc-name` argument to define the name of the service. |

**Example:**

```
Router(config)# sbc mysbc
```

| **Step 3** sbe | Enters the mode of an SBE entity within an SBC service. |

**Example:**

```
Router(config-sbc)# sbe
```

| **Step 4** sip method-profile profile-name | Configures a method profile. If you enter the `profile-name default`, the default profile is configured. This profile is used for all adjacencies that do not have a specific profile configured. |

**Example:**

```
Router(config-sbc-sbe)# sip method-profile profile1
```

| **Step 5** method name | Adds a method with the specified name to the profile. This field can be 1 to 32 characters (inclusive) in length and is case-insensitive. The `no` form of this command deletes the method with that name from the profile. |

**Example:**

```
Router(config-sbc-sbe-sip-mth)# method test
```

| **Step 6** map-status-code | Enters the SIP method profile element configuration mode. |

**Example:**

```
Router(config-sbc-sbe-sip-mth-ele)# map-status-code
```

| **Step 7** range statuscoderange value statuscodevalue | Maps a range of response codes to a response code. |

**Example:**

```
Router(config-sbc-sbe-sip-mth-ele-map)# range 5XX value 500
```

| **Step 8** end | Exits the method profile mode and returns to Privileged EXEC mode. |

**Example:**

```
Router(config-sbc-sbe-sip-mth-prf)# end
```
Applying Response Code Mapping

Apply response code mapping by associating it with an adjacency.

**SUMMARY STEPS**

1. configure terminal
2. sbc sbc-name
3. sbe
4. adjacency sip adjacency-name
5. method-profile inbound profile-name
6. end
7. show sbc sbc-name sbe sip method-profile name

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td>Example: Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enters the mode of an SBC service.</td>
</tr>
<tr>
<td>Example: Router(config)# sbc mysbc</td>
<td>Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td><strong>Step 3</strong> sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td>Example: Router(config-sbe)# sbe</td>
<td></td>
</tr>
</tbody>
</table>
**SIP Message Editing Using Profiles**

**Step 4**  
**Command or Action**: `adjacency sip adjacency-name`  
**Example**:  
```
Router(config-sbc-sbe)# adjacency sip test
```
**Purpose**: Enters the mode of an SBE SIP adjacency. Use the `adjacency-name` argument to define the name of the service.

**Step 5**  
**Command or Action**: `method-profile inbound profile-name`  
**Example**:  
```
Router(config-sbc-sbe-adj-sip)# method-profile inbound profile1
```
**Purpose**: Sets `profile1` to be used for inbound signaling on adjacency test.  
**Note**: When attaching a method profile to an adjacency, the adjacency must be in the “no attach” state.

**Step 6**  
**Command or Action**: `end`  
**Example**:  
```
Router(config-sbc-sbe-adj-sip)# end
```
**Purpose**: Exits the header profile mode and returns to Privileged EXEC mode.

**Step 7**  
**Command or Action**: `show sbc sbc-name sbe sip method-profile name`  
**Example**:  
```
Router# show sbc mysbc sbe sip method-profile one
```
**Purpose**: Displays the header profile information.

---

**Header Profiles**

Header profiles processing occurs in a two-stage process. In the first stage, the following steps occur:

1. Select next header from the message.
2. Look through the header profile for rules affecting the selected header.
3. In configured order, try to apply each rule to the header.
4. If the action is to add a header, then ignore this rule and move on to the next.
5. If the match condition is FALSE then move onto the next rule, do not evaluate any parameter profile.
6. Apply the action or parameter profile described in the element. If this is to remove the header, then move on to the next header in the message.

The second stage adds new headers to the message. Because it occurs after the first stage, there is a well-defined group of headers in the message. The steps are:

1. Take each rule that adds a header to the message.
2. If the action is to add the first instance of the header only and there is already a header with that name in the message, then move onto the next addition rule.

**Note**: If another action has replaced the name of header then it is the replaced name that is used to test whether a new header should be added. That is, any header-name replacements performed in stage 1 are used in this stage of header-name comparisons, and not the original header-names from the arriving message.

3. Add the header if the match condition evaluates to TRUE.
4. Apply any rules defined for that header in user-configured order with this name. Only apply rules that are ordered after the add header rule, if the header was added.

---

**Command or Action** | **Purpose**
---|---
`adjacency sip adjacency-name` | Enters the mode of an SBE SIP adjacency. Use the `adjacency-name` argument to define the name of the service.

`method-profile inbound profile-name` | Sets `profile1` to be used for inbound signaling on adjacency test.  
**Note**: When attaching a method profile to an adjacency, the adjacency must be in the “no attach” state.

`end` | Exits the header profile mode and returns to Privileged EXEC mode.

`show sbc sbc-name sbe sip method-profile name` | Displays the header profile information.
This section contains the following topics:

- Restrictions for Configuring Header Profiles, page 23-17
- Information About Header Profiles, page 23-17
- Header Manipulation, page 23-18
- Header Profile Configuration Information, page 23-25
- Configuring Header Profiles, page 23-25
- Applying Header Profiles, page 23-27

Restrictions for Configuring Header Profiles

Review the following restrictions for header profiles:

- Any given profile must be exclusively a whitelist or a blacklist.
- Two profiles are applied to process any given SIP message: one inbound and, if permitted through that, one outbound.
- SIP headers that are essential to the operation of Cisco Unified Border Element (SP Edition) cannot be blacklisted and are implicitly added to any whitelist.
- Profiles cannot be removed while they are in active use by an adjacency.
- For provisional filtering, provisional responses may not be blocked where the sender has required reliable provisional responses (SIP 100rel). This is to ensure that Cisco Unified Border Element (SP Edition) does not interfere with the call setup (as per RFC3262) by dropping the provisional response.
- Header profile conditional matching can be performed against any part of the message. The matches can be exact matches or even sub-strings of any given field.
- The conditions may be associated with a specific header referenced by the header profile header definition, but can also reference other non-vital parts of the message in order to evaluate the conditional expression; thus the condition could be associated with header P-Asserted-Identity while checking against the contents of the Call-Info header.

Information About Header Profiles

After you configure a profile, you can assign it for a default application. Any SIP adjacency can apply it to signaling for that adjacency.

You can add or remove headers from profiles at any time. Headers configured on a profile must contain characters that are valid for a SIP header.

Profiles cannot be deleted while any adjacency is using them. You can see which adjacencies are using a profile by entering the following show command:

```
show sbc sbc-name sbe sip method-profile [profile-name]
```

or

```
show sbc sbc-name sbe sip essential-methods
```

The following are the essential SIP headers, which must not be configured on any profile:

- Allow
- Authorization
- Call-ID
SIP Message Editing Using Profiles

- Contact
- Content-Length
- Content-Type
- CSeq
- Event
- Expires
- From
- Max-Forwards
- Min-Expires
- Proxy-Authenticate
- Proxy-Authorization
- Proxy-Require
- Record-Route
- Referred-By
- Referred-To
- Replaces
- Require
- Route
- Subscription-State
- Supported
- To
- Via
- WWW-Authenticate

Note: Profiles are an optional part of the configuration. If no profile is applicable to a given SIP signal, then the essential headers are processed and all other headers are not forwarded.

Header Manipulation

You can modify non-essential headers in SIP messages using header and parameter profiles. The following information summarizes the supported actions:

- Pass the header unchanged (whitelist functionality).
- Conditionally pass the header unchanged.
- Remove the header (blacklist functionality).
- Conditionally remove the header.
- Replace the name of the header. The replacement name cannot be that of a vital header.
- Conditionally replace the header content (appearing after the “:”).
- Add a new instance of a header to a message regardless of whether or not the header already exists.
- Add the first instance of the header to the message, if a header with this name does not already exist.
Chapter 23  SIP Message Manipulation

SIP Message Editing Using Profiles

- A combination of the above actions can be specified as a set or group of actions to be performed within a profile.
- The header profiles can be used in method profiles to allow header actions only associated with specific requests types.
- Parameter profiles can be associated with headers in header profiles.
- Header content can be stored in variables and later expanded during replace-value actions.
- Privacy headers are treated as unknown headers, which by default would be blacklisted (stripped). However, the SBC can be configured to pass through SIP Privacy headers.
- Regular expression matching can be performed on headers.

You can match against any part of a header but only replace the entire header. A parameter profile extends this capability to change individual SIP URI parameters associated with a header. Header profiles can only modify non-vital header information. To display the vital header information, use the `show sbc test sbe sip essential-method`, `show sbc test sbe sip essential-headers`, or `show sbc test sbe sip essential-parameters` commands.

Parameter profiles can be specified to match the following parts of the message.

- Request URI
- To
- From
- Contact

To modify the parameters in the Request-line, associate a parameter profile with a method profile. To modify the parameters in the Contact, To, or From headers, associate a parameter profile in the header profile.

---

Event Header in Publish Method

As per RFC3903, the SIP PUBLISH request must contain an Event header. In releases earlier than Cisco IOS XE Release 3.2S, the SBC could pass through the PUBLISH method using the existing message manipulation framework, but could not pass through the Event header. The effect of this was that attempts to use the PUBLISH services (containing an Event header) through the SBC were blocked.

From Cisco IOS XE Release 3.2S, the SBC can pass through the PUBLISH method containing Event header using the existing message manipulation framework. Preset header manipulations accessed by inherit-profiles are modified to pass-on the Event header.

The Event Header in Publish Method feature does not affect the behaviors for SUBSCRIBE, REFER, and NOTIFY methods. Event headers are passed through unchanged. For all the other methods, the Event header is treated generically.

---

Header Profile Conditional Matching

To allow header manipulation, a set of conditions can be specified in order to dictate the rules under which the header actions will be applied. Conditional matching allows comparisons to be performed against any part of the message. The matches can be exact matches or even sub-strings of any given field.

The conditions can be associated with a specific header referenced by the header profile header definition, but equally can also reference other non-vital parts of the message in order to evaluate the conditional expression.
Absence of a condition (conditional expression) implies the condition for the action is always true.

Each condition represents a part of the message to be manipulated, and the operation to be performed. A condition can be defined in the following ways:

condition comparison-type operator comparison-value

or

condition boolean-operator operator {true | false}

Example:
condition header-value contains "Cisco"
condition is-request eq true

Table 23-1 lists the comparison types.

Table 23-1 Comparison Types

<table>
<thead>
<tr>
<th>Comparison Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status-code</td>
<td>Response code value</td>
</tr>
<tr>
<td>header-value</td>
<td>Current header content</td>
</tr>
<tr>
<td>header-name name</td>
<td>Content of a different header</td>
</tr>
<tr>
<td>variables</td>
<td>Match on variable content</td>
</tr>
<tr>
<td>adjacency</td>
<td>Match on adjacency settings</td>
</tr>
<tr>
<td>transport</td>
<td>Match on transport addresses or ports</td>
</tr>
<tr>
<td>header-uri</td>
<td>Match on parts of the URI (username)</td>
</tr>
<tr>
<td>request-uri</td>
<td>Match on parts of the request-URI (username)</td>
</tr>
<tr>
<td>word</td>
<td>Match on static strings</td>
</tr>
</tbody>
</table>

Table 23-2 lists the operators.

Table 23-2 Operators

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[not] eq</td>
<td>Equals or not equal</td>
</tr>
<tr>
<td>[not] contains</td>
<td>Contains or does not contain</td>
</tr>
<tr>
<td>[not] regex-match</td>
<td>Regular expression matching (BRE)</td>
</tr>
<tr>
<td>store-as</td>
<td>Store rules only</td>
</tr>
</tbody>
</table>

Table 23-3 lists the boolean operators.

Table 23-3 Boolean Operators

<table>
<thead>
<tr>
<th>Boolean Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-sip-uri</td>
<td>Does the header contain a sip: URI</td>
</tr>
<tr>
<td>is-tel-uri</td>
<td>Does the header contain a tel: URI</td>
</tr>
</tbody>
</table>
Chapter 23      SIP Message Manipulation

The following restrictions apply for conditional matching:

- Multiple conditional expressions against the same header can be added each containing unique actions and conditions to build complex manipulations.
- Each condition must be entered one at a time. To add a subsequent condition to an existing condition, the condition must begin with “and” or “or”. If the condition does not contain “and” or “or”, it effectively overwrites any conditions already defined.
- If no profile-type is explicitly expressed in the header profile command line definition then the assumed header profile type will be “whitelist”.
- Multiple headers of the same type can be declared in any one profile defining either different action types or conditions.
- Character “*” can be used as a wildcard header, although only one wildcard header entry can be configured per profile.
- Duplicate header names with differing actions or conditions can be identified with the “entry <integer>” parameter in the command line. This can be used for the purposes of editing or deletion of a specific action related to a header. If no “entry” in the command line then it is assumed that the first entry related to the header of this header type is being configured.

Store Rules Declaration

The data extracted from headers can be stored into variables. The store rules are defined which are executed prior to any header element actions. Store rules are specialized header elements of the format:

```
Store-Rule:<entry>
```

The store rules contain conditions which allow storage in one of the following two ways:

1. A condition can contain a “store-as” keyword to directly store a string or complete header value into a variable.
   ```
   condition comparison-type store-as variable-name
   ```
   Example:
   ```
   condition header-value store-as var1
   ```
   The content of header-value will be stored into var1.

2. A regular expression can be applied to a header using keyword “regex-match”. If the regular expression contains one or more (up to five max) sets of escaped parentheses ‘(\)’ around specific parts of the regular expression, then if the regular expression successfully matches, the values of each parts of the match grouped by the parentheses are extracted and stored into variables defined in the regex-match keyword arguments.
   ```
   condition comparison-type regex-match [store-as variable-name....(up to 5)]
   ```
   Example:
   ```
   condition header-name P-Asserted-Identity header-value regex-match sip:\((.*)@([Cc]isco.com store-as var1
   ```

### Table 23-3     Boolean Operators

<table>
<thead>
<tr>
<th>Boolean Operators</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>is-request</td>
<td>Is the message a request</td>
</tr>
<tr>
<td>is-100rel-required</td>
<td>Is the call performing 100rel</td>
</tr>
<tr>
<td>is-defined</td>
<td>Test if a variable is defined</td>
</tr>
</tbody>
</table>
For the complete list of comparison types, operators, and boolean operators, refer Table 23-1, Table 23-2, and Table 23-3.

Extracted variables can later be used in the actions which require values such as replace-value, add-first-header/add-header. Variables are expanded by use of “${var}” format within the replacement string.

Request Line Modification

You can perform limited modification to the request-line with action replace-value in header profiles. The use of the request-line forming part of the header profiles is the preferred method for changes (including parameter profiles) to the request-line.

The format of the value used in action replace-value is:

```
sip:user@host[:port]
```

The variables that are already extracted to the store rules can be used in the construction of the Request Line.

Example:

```
"sip:$(user)@${host}"
```

Request-line is a specialized header element of the format:

```
Request-URI:<entry>
```

Note

Changes to the request-line must meet the SIP RFC 3261 formatting rules, and any host declared in the replacement must be a valid host to the SBC. User configuration cannot pre-screen the configured changes due to the possibility of variables being present in the configured replacement value. It is only at run-time when the actual request-line can be determined, and errors in request-line construction can result in call failures. Extreme care must be taken when using this feature to prevent call failures.

Parse User Name Parameters

You can configure the SBC to search and parse SIP and SIPS URIs for user name parameters in messages received on an adjacency. If the SIP and SIPS URIs contain any user name parameters, those parameters are treated as regular URI parameters. This is applicable to SIP and SIPS URIs within the Request URI, and the To and From headers for INVI TE requests and out-of-dialog requests.

The following is an example of a URI with a username parameter:

```
“sip:username;cic=1234@host.com;user=phone”
```

Here, ‘cic=1234’ is treated as a URI parameter, such as ‘user=phone’, and the username is taken to be ‘username’, instead of ‘username;cic=1234’

Use the command `uri_username parameters parse` to enable parsing.

Suppress Expires Header

You can configure the SBC to suppress the Expires Header in the outgoing INVITE requests. Use the command `header-name expires suppress` to remove the Expires Header.
Configuring Customer P-Asserted-Identity

You can configure the SBC to specify a value for the P-Asserted-Identity on the outgoing SIP message. The header is added to all requests and responses except ACK, CANCEL, INFO, PRACK, REGISTER and UPDATE.

Use the `header-name p-asserted-id [header-value [header-value] | assert]` command to specify a value for the P-Asserted-Identity.

SIP Destination ID

**Note**

This feature is applicable only to the INVITE and non-REGISTER out-of-dialogue requests.

When routing a call, the destination address or called party identity is typically derived from the Request URI. However, there are other headers where this information could potentially be derived from, such as To: or P-Called-Party-ID.

You can define an ordered list of headers that can be used to derive the called party address. The headers can include any non-essential SIP header, or To:, and Request URI. A maximum of ten headers can be configured in a header list. The header with priority 1 is analyzed first, the header with priority 2 is analyzed next, and the header with priority 10 is analyzed last.

The following sections describe how this feature works on incoming and outgoing requests.

Incoming Requests

For incoming requests:

- By default, the SBC extracts the called party identity from either the P-Called-Party-ID: header or from the Request URI.
- If the SBC finds multiple instances of a given header in a received SIP message, the first instance is used for called party identity extraction. If the SBC encounters a syntax error while extracting the identity, the SBC creates a log, and moves to the next header in the priority list.
- If a header is not present in the SIP request, or if a header in the header list contains a SIP URI without a username, the SBC moves to the next header in the header list.
- After all headers have been tried without success, the SBC extracts the called party identity from the Request URI.
- The header list may include the Request URI to enable the SBC to look for the called party identity from the Request URI when it gets to a point where the Request URI is prioritized in the list. If the list contains only the Request URI, the SBC looks at only the Request URI.

Outgoing Requests

By default, the SBC reinserts both the domain and the username from the called party identity back into the SIP header from which the identifier originally came on the inbound side.

Outgoing Request URI:

- If the called party identity was originally extracted from the Request URI, the Request URI is reconstructed using the called party identity.
- If the called party identity was originally extracted from another header, the username and domain in the Request URI from the SIP message received are preserved. This is done before any SIP header filtering or other editing function (for example, IP/FQDN URI translation) is applied to the Request URI.

Outgoing To Header or Passed Through Arbitrary Header:

- If the called party identity was originally extracted from a header (rather than the Request URI) and that header has been passed through using the inbound adjacency's header manipulation functionality, the SBC inserts the domain and username back into the header, thereby preserving the scheme, URI parameters, and header parameters that were in the original message. Failures due to corruption of header because of the inbound header filtering configuration are logged by the SBC, but other failures are ignored.

- The called party identity may have been edited by the SBC (for example, as part of Number Manipulation) before being reinserted into the outgoing message. This is done only for the first instance of the header in the outbound SIP request before any outbound header filtering or any other editing is applied to the header. There is no restriction on header filtering. You may configure the header editing rules that may subsequently remove or change the header containing the called party identity.

- We recommend that you configure action pass on the inbound header filter profile for all the headers specified in the header list. These headers can then be filtered by the outbound header filter profile.

To configure the destination address header list, use the `dst-address` and `header-priority` commands. See the $paranum>Configuring an Ordered List of Headers for Deriving the SIP Destination Address? section on page 23-28 for details on configuring header-priority for deriving SIP source ID.

The SBC can be configured to perform conditional matching based on these derived values. See the $paranum>Header Profile Conditional Matching? section on page 23-19 for more details.

**SIP Source ID**

When routing a call, the source number can be analyzed and modified using a call policy. The source address is typically derived from the From: header. There are, however, other headers from where this information could potentially be derived from, such as P-Preferred-Identity, P-Asserted-Identity, Remote-Party-ID.

You can define an ordered list of headers that can be used to derive the called party address. The headers can include any non-essential sip header and the From header. The SIP Source ID feature also enables you to derive the source number from an ordered set of headers for the calls that were either redirected or diverted. A maximum of ten headers can be configured in the header list. The header with priority 1 is analyzed first, header with priority 2 is analyzed next and the header with priority 10 is analyzed last.

To configure the source address header list, use the `src-address` command and the `header-priority` command.

See the $paranum>Configuring an Ordered List of Headers for Deriving SIP Source Address? section on page 23-30 for details on configuring header-priority for deriving SIP source ID.

**SIP Source ID for Diverted Calls**

For diverted calls, you can use the address of the party that diverted the call to derive the source address for source analysis. All the diverted calls contain a Diversion: header that contains the details of the party that diverted the call. The SBC can be configured to enter a list of headers for the diverted calls, from which the source number can be derived.
For Cisco IOS XE Release 3.1.0S, this list can only contain one Diversion: header.

To configure the source address header list, use the `div-address` command and the `header-priority` command.

See the "Configuring an Ordered List of Headers for Deriving SIP Source Address of Diverted Calls? section on page 23-31 section for details on configuring header-priority for deriving SIP source ID for diverted calls.

The SBC can be configured to perform conditional matching based on these derived values. See the "Header Profile Conditional Matching? section on page 23-19 for more details on conditional matching.

### Header Profile Configuration Information

Consideration needs to be given as to the effect of an action or set of actions in conjunction with the default profile behavior (whitelist/blacklist).

An empty blacklist will effectively try to pass on any non-vital header.

An empty whitelist will effectively drop all non-vital headers.

The behavior becomes more complex when conditions are associated with headers.

It is important to consider what actions are defined on the in-bound side. If an empty whitelist header profile is associated with the in-bound side, then no non-vital headers will be visible at all to the outbound side, and therefore, actions applied to the out-bound sides profile may appear not to work. You may need to consider adding actions to ‘pass’ a specific header on the in-bound side by adding the header to a whitelist (with action as-profile or pass) or adding the header with action ‘pass’ in a blacklist.

For example, if a header profile is defined as a whitelist (default behavior), and a header action to modify the header-value is inserted with a condition, then the action will be processed if the condition is TRUE and the header modified, but will be ignored if the condition is FALSE.

Because the header is inserted into the whitelist it might well be assumed that it would be passed on unmodified if the condition is FALSE, however, if the condition is FALSE, the action (entry) is ignored, and therefore it is as if the header is not present in the whitelist so the header will not be passed on.

To overcome this, a second entry with action ‘pass’ can be entered; thus if the headers condition is TRUE, the content with be modified, but if the condition is false, it will be ignored and continue to process any other entries. The second entry has an action ‘pass’ and will cause the header to be passed on.

### Configuring Header Profiles

This procedure shows how to configure header profiles.

**SUMMARY STEPS**

1. `configure terminal`
2. `sbc sbc-name`
3. `sbe`
4. `sip header-profile profile-name`
5. `blacklist`
6. `description text`
7. header name [entry number]
8. action {add-first-header | add-header | as-profile | drop-msg | pass | replace-name | replace-value | strip}
9. condition [comparison-type | boolean-operator | operator | comparison-value]
10. end
11. show sbc sbc-name sb e sip header-profile [profile-name]
12. show sbc sbc-name sb e sip essential-headers

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enters the submode for configuring the header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td>Router(config)# sbc mysbc</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip header-profile profile-name</td>
<td>Configures a header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>If you enter the profile-name default, the default profile is configured. This profile is used for all adjacencies which do not have a specific profile configured.</td>
</tr>
<tr>
<td>Router(config-sbc-sbe)# sip header-profile profile1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> blacklist</td>
<td>Configures a profile to be a blacklist.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>The no form of this command configures the profile to be a whitelist. Note By default, profiles are whitelists.</td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-hdr)# blacklist</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> description text</td>
<td>Adds a description for the specified profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>The no form of this command removes the description. This description is displayed when the show command is used for this profile and is displayed for each profile when displaying a summary of all profiles.</td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-hdr)# description blacklist profile</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> header name [entry number]</td>
<td><strong>header name</strong>—Configures the SIP header that will be modified. Enters SBC SBE SIP-HDR-ELE configuration mode. <strong>entry number</strong>—Specifies which action entry to work on.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc-sbe-sip-hdr)# header</td>
<td></td>
</tr>
<tr>
<td>Organization entry 1</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 23  SIP Message Manipulation

SIP Message Editing Using Profiles

**Command or Action**

| Step 8 | action {add-first-header | add-header | as-profile | drop-msg | pass | replace-name | replace-value | strip} |
|--------|-------------------------------------------------|
| Purpose | Specifies the type of action to be applied to the header. In the example, the action specified is to conditionally replace the header content with a replace value of XYZcompany. |

**Example:**

Router(config-sbc-sbe-sip-hdr-ele)# action replace-value XYZcompany

| Step 9 | condition [comparison-type | boolean-operator | operator | comparison-value] |
|--------|-------------------------------------------------|
| Purpose | Specifies the condition to match before taking an action to a SIP message profile. If the condition is met, the action specified in step 8 is performed. Enters SIP header profile configuration mode. In the example, the value of the condition header-value is ABCCompany, which is matched and thus the value ABCCompany is replaced with XYZcompany. |

**Example:**

Router (config-sbc-sbe-sip-hdr-ele-act)# condition header-value ABCcompany

<table>
<thead>
<tr>
<th>Step 10</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Exits the SBC SBE SIP-HDR-ELE configuration mode and returns to Privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Example:**

Router(config-sbc-sbe-sip-hdr-ele)# end

<table>
<thead>
<tr>
<th>Step 11</th>
<th>show sbc sbc-name sbe sip header-profile [profile-name]</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Displays details for the header profile with the designated name. Use the profile-name default to view the default profile. Displays a list of all configured method profiles if no profile-name is specified.</td>
</tr>
</tbody>
</table>

**Example:**

Router# show sbc mysbc sbe sip header-profile profile1

<table>
<thead>
<tr>
<th>Step 12</th>
<th>show sbc sbc-name sbe sip essential-headers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Purpose</td>
<td>Displays a list of the essential headers.</td>
</tr>
</tbody>
</table>

**Example:**

Router# show sbc mysbc sbe sip essential-headers

---

### Applying Header Profiles

This procedure shows how to apply header profiles.

**SUMMARY STEPS**

1. configure
2. sbc sbc-name
3. sbe
4. adjacency sip adjacency-name
5. header-profile inbound profile-name
6. end
7. show sbc sbc-name sbe sip header-profile name
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>configure</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router# configure</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>sbc sbc-name</td>
<td>Enters the mode of an SBC service. Use the <code>sbc-name</code> argument to define the name of the service.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config)# sbc mySBC</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>adjacency sip adjacency-name</td>
<td>Enters the mode of an SBE SIP adjacency. Use the <code>adjacency-name</code> argument to define the name of the service.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe)# adjacency sip sipGW</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>header-profile inbound profile-name</td>
<td>Sets the inbound header profile to be used for inbound signaling on adjacency sipGW. <strong>Note</strong>: When attaching a header profile to an adjacency, the adjacency must be in the “no attach” state.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe-adj-sip)# header-profile inbound profile1</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>end</td>
<td>Exits the SBE SIP adjacency mode and returns to Privileged EXEC mode.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe-adj-sip)# end</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>show sbc sbc-name sbe sip header-profile name</td>
<td>Displays the header profile information.</td>
</tr>
<tr>
<td></td>
<td>Example:</td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router# show sbc sbc-name sbe sip header-profile name</td>
<td></td>
</tr>
</tbody>
</table>

## Configuring an Ordered List of Headers for Deriving the SIP Destination Address

This task configures a list of headers for deriving SIP destination address.

### SUMMARY STEPS

1. configure terminal
2. sbc `sbc-name`
3. sbe
4. sip header-profile `profile-id`
5. dst-address
6. `header-prio 1 header-name header-name`
7. `header-prio 2 header-name header-name`
8. `header-prio 3 header-name header-name`
9. `end`
10. `show sbc sbc-name sbe sip header-profile profile-id`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>configure</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong>sbc <code>sbc-name</code></td>
<td>Enables entry into the mode of an SBC service. Use the <code>sbc-name</code> argument to define the name of the SBC.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# sbc mySbc</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong>sbe</td>
<td>Enables entry into the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc)# sbc mySbc sbe</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong>sip header-profile</td>
<td>Creates the SIP header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe)# sip header-profile Hprof1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong>dst-address</td>
<td>Enables entry into the mode to configure destination address.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-sip-hdr)# dst-address</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong><code>header-prio 1 header-name header-name</code></td>
<td>Configures the header priority, and specifies the header to be used.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-sip-hdr-dst)# header-prio 1 header-name P-Called-Party-ID</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong><code>header-prio 2 header-name header-name</code></td>
<td>Configures the header priority, and specifies the header to be used.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-sip-hdr-dst)# header-prio 2 header-name To</td>
<td></td>
</tr>
<tr>
<td><strong>Step 8</strong><code>header-prio 3 header-name header-name</code></td>
<td>Configures the header priority, and specifies the header to be used.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-sip-hdr-dst)# header-prio 3 header-name Request-uri</td>
<td></td>
</tr>
</tbody>
</table>
Chapter 23      SIP Message Manipulation

SIP Message Editing Using Profiles

Configuring an Ordered List of Headers for Deriving SIP Source Address

This task configures a list of headers for deriving SIP source address.

**SUMMARY STEPS**

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip header-profile profile-id
5. src-address
6. header-prio 1 header-name header-name
7. header-prio 2 header-name header-name
8. header-prio 3 header-name header-name
9. end
10. show sbc sbc-name sbe sip header-profile profile-id

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td>Example: configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enables entry into the mode of an SBC service. Use the sbc-name argument to define the name of the SBC.</td>
</tr>
<tr>
<td>Example: Router(config)# sbc mySbc</td>
<td></td>
</tr>
</tbody>
</table>
### Configuring an Ordered List of Headers for Deriving SIP Source Address of Diverted Calls

This task configures a list of headers for deriving SIP source address of diverted calls.

**SUMMARY STEPS**

1. configure terminal
Chapter 23  SIP Message Manipulation

SIP Message Editing Using Profiles

2.  `sbc sbc-name`
3.  `sbe`
4.  `sip header-profile profile-id`
5.  `div-address`
6.  `header-prio 1 header-name header-name`
7.  `end`
8.  `show sbc sbc-name sbe sip header-profile profile-id`

## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong>  &lt;br&gt; <code>configure</code></td>
<td>Enables the global configuration mode.</td>
</tr>
</tbody>
</table>
| **Example:**  
  `Router# configure` | |
| **Step 2**  <br> `sbc sbc-name` | Enables entry into the mode of an SBC service.  
  *Use the `sbc-name` argument to define the name of the sbc.* |
| **Example:**  
  `Router(config)# sbc mySbc` | |
| **Step 3**  <br> `sbe` | Enables entry into the mode of an SBE entity within an SBC service. |
| **Example:**  
  `Router(config-sbc)# sbe mySbc sbe` | |
| **Step 4**  <br> `sip header-profile` | Creates SIP header profile. |
| **Example:**  
  `Router(config-sbc-sbe)#sip header-profile Hprof1` | |
| **Step 5**  <br> `div-address` | Enables entry into the mode to configure source address for diverted calls. |
| **Example:**  
  `Router(config-sbc-sbe-sip-hdr)# div-address` | |
| **Step 6**  <br> `header-prio 1 header-name header-name` | Configures the header priority, and specifies the header to be used. |
| **Example:**  
  `Router(config-sbc-sbe-sip-hdr-src-div)# header-prio 1 header-name Diversion` | |
SIP Message Editing Using Profiles

Following is an example for the `show` command output after the header list—for destination address, source address, and diversion address—is configured on SBC:

```
ASR-1002# show sbc mine sbe sip header-profile Hprof1
Header profile "Hprof1"
Description:
Type: Whitelist
dst-address: (inbound only)
  header-prio 1 header-name P-called-ID
  header-prio 1 header-name To
  header-prio 1 header-name Request-uri
src-address: (inbound only)
  header-prio 1 header-name Remote-Party-ID
  header-prio 2 header-name P-Preferred-Identity
  header-prio 3 header-name From
div-address (inbound only)
  header-prio 1 Diversion
store-rules:
  No store-rule entries found.
request-line:
  No request-line entries found.
headers:
  test
  entry 1
  description:
    action add-first-header value "cisco"
    condition is-request eq true
Not in use with any adjacencies
Not in use with any method-profile
```

Provisional Response Filtering

Provisional response filtering makes it possible to block 1XX responses (except 100) sent by endpoints. When configuring provisional response filtering, keep the following in mind:

- Provisional responses may not be blocked where the sender has required reliable provisional responses (SIP 100rel).
- Dropping responses where 100_rel is required is not recommended. It may prevent call setup since RFC3262 states subsequent responses should not be sent.
Note
A call attempted with the "Required: 100Rel" header in the INVITE will fail when the adjacency is configured with a header profile to drop 183 messages.

This section contains the following topics:
- Provisional Response Filtering Information, page 23-34
- Configuring Provisional Response Filtering, page 23-34
- Applying Provisional Response Filtering, page 23-35

Provisional Response Filtering Information

Provisional response filtering is achieved by the use of the action drop-msg command. The action must be associated with the wildcard header action *. A condition should be added to match on the specific response code that must be dropped.

Note
The header action * can only be used one time in a profile.

Configuring Provisional Response Filtering

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip header-profile profile-name
5. header *
6. action drop-msg
7. condition status-code
8. end

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td></td>
</tr>
<tr>
<td>configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td></td>
</tr>
<tr>
<td>sbc sbc-name</td>
<td>Enters the submode for configuring the header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td>Router(config)# sbc mysbc</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step 3</th>
<th>sbe</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc)# sbe</td>
</tr>
</tbody>
</table>

Enters the mode of an SBE entity within an SBC service.

<table>
<thead>
<tr>
<th>Step 4</th>
<th>sip header-profile profile-name</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc-sbe)# sip header-profile profile1</td>
</tr>
</tbody>
</table>

Configures a header profile.

If you enter the **profile-name default**, the default profile is configured. This profile is used for all adjacencies which do not have a specific profile configured.

<table>
<thead>
<tr>
<th>Step 5</th>
<th>header *</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc-sbe-sip-hdr)# header *</td>
</tr>
</tbody>
</table>

Configures a profile to be a blacklist.

The no form of this command configures the profile to be a whitelist.

**Note**  By default, profiles are whitelists.

**Note**  In order to filter provisional responses always use the asterisk (*) as the header name with the **header** command as shown in the command example.

<table>
<thead>
<tr>
<th>Step 6</th>
<th>action drop-msg</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc-sbe-sip-hdr-ele)# action drop-msg</td>
</tr>
</tbody>
</table>

Configures the action to take on an element type in a header.

<table>
<thead>
<tr>
<th>Step 7</th>
<th>condition status-code</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc-sbe-sip-hdr-ele-act)# condition status-code eq 183</td>
</tr>
</tbody>
</table>

Specifies a condition to match before taking an action to a SIP message profile.

<table>
<thead>
<tr>
<th>Step 8</th>
<th>end</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Example:</strong></td>
<td>Router(config-sbc-sbe-sip-hdr-ele-act)# end</td>
</tr>
</tbody>
</table>

Returns to privileged EXEC mode.

### Applying Provisional Response Filtering

This procedure shows how to apply provisional response filtering.

### SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. adjacency sip adjacency-name
5. header-profile inbound profile-name
6. end
7. show sbc sbc-name sbe sip header-profile name
DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enters the mode of an SBC service. Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# sbc mysbc</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> adjacency sip adjacency-name</td>
<td>Enters the mode of an SBE SIP adjacency. Use the adjacency-name argument to define the name of the service.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe)# adjacency sip sipGW</td>
<td></td>
</tr>
<tr>
<td><strong>Step 5</strong> header-profile inbound profile-name</td>
<td>Sets the inbound header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-adj-sip)# header-profile inbound profile1</td>
<td></td>
</tr>
<tr>
<td><strong>Step 6</strong> end</td>
<td>Exits the SBE SIP adjacency mode and returns to Privileged EXEC mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-adj-sip)# end</td>
<td></td>
</tr>
<tr>
<td><strong>Step 7</strong> show sbc sbc-name sbe sip header-profile name</td>
<td>Shows details of the specified SIP header profile.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router# show sbc MySbc sbe sip header-profile profile1</td>
<td></td>
</tr>
</tbody>
</table>

Parameter Profiles

Parameter profiles allow you to specify specific URI parameter names and allow the removal, replacement, or the addition of specific non-vital URI parameters within certain headers.

The header profile allows potential conditional matching against SIP URI parameters forming part of a limited set of headers. It only allows complete replacement of the header and or content.

The parameter profile will allow actions to be performed only on the SIP URI parameters and not header parameters.
This section contains the following topics:

- Restrictions for Configuring Parameter Profiles, page 23-37
- Information About Parameter Profiles, page 23-37
- Configuring Parameter Profiles, page 23-38
- Applying a Parameter Profile to a Header Profile, page 23-39
- Associating with an Adjacency, page 23-41

Restrictions for Configuring Parameter Profiles

Review the following restrictions for parameter profiles:

- A parameter profile is only permitted to act on parameters associated with SIP URIs and not header parameters.
- To prevent call processing failures, actions cannot be performed against vital (essential) parameters.
- Parameter profiles work only on the outbound side.
- Some of the existing adjacency settings may impact the way parameter actions are affected. For example, consider the adjacency setting Rewrite to Header is set as follows:

```
sbc test
  sbe
      adjacency sip <adj name>
        passthrough [to/from]
```

This setting can cause the To: and or From: headers to be passed from inbound to outbound side. The default setting on an adjacency, however, is FALSE (no “passthrough [to/From]” appears in the show run against the adjacency) which means that the To: and From: headers are effectively always re-written on the outbound side by default. The impact of this is that parameter profiles actions applied to the inbound sides To: and/or From: headers will be lost on the outbound side unless ‘passthrough [to/from]’ is set in the configuration. Thus the action add-not-present can look like it always adds a parameter on the outbound side, even when the parameter is present on the in-bound side.

- If a parameter profile adds a parameter to the request-line, and the To: header does not have setting ‘passthrough to’ set against the adjacency, then the re-writing of the To: header which is typically based on the Request Line, will cause the parameter to also appear in the To: header.
- The content of the Request-line may affect the behavior of parameter profiles attached to method profiles. If the request-line that arrives on the in-bound side of the call directly addresses the address of Cisco Unified Border Element (SP Edition), then effectively any call that originates on the out-bound side requires a new Request Line to be generated. This means that parameters arriving on the in-bound side are effectively lost and can cause the action add-not-present to look like it always adds a parameter.
  
    If however, the Request Line address the final destination, then the Request Line is effectively passed across to the outbound side and modified as needed. Parameters in this case are visible on the out-bound side.

Information About Parameter Profiles

Parameter profiles form a set of actions that can be performed against any one header or request-line. Parameter profiles can only be specified against the following parts of the message:
SIP Message Editing Using Profiles

- Request URI
- To
- From
- Contact

To modify parameters in Contact, To, or From headers, associate a parameter profile in the header profile.

To modify parameters in the request-line, associate a parameter profile with a method profile.

Note: Parameter profiles can be associated with essential methods even though method profiles are not allowed to blacklist/whitelist essential methods.

Configuring Parameter Profiles

Perform this task to configure parameter profiles.

SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip parameter-profile {profile-name}
5. parameter {parameter name}
6. action {add-not-present | add-or-replace | strip}
7. end
8. show sbc sbc-name sbe sip-parameter-profile [profile name]
9. show sbc sbc name sbe sip essential-parameters

DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enters the mode of an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td>Router(config)# sbc mysbc</td>
<td></td>
</tr>
<tr>
<td><strong>Step 3</strong> sbe</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
</tbody>
</table>
### Command or Action | Purpose
---|---
**Step 4**  
`sip parameter-profile (profile-name)` | Configures a parameter profile and enters SBE SIP header configuration mode.
  
**Example:**  
Router(config-sbc-sbe)# sip parameter-profile parmprof

**Step 5**  
`parameter (parameter name)` | Adds a parameter with a specified name to the parameter profile.
  
**Example:**  
Router(config-sbc-sbe-sip-prm)# parameter user

**Step 6**  
`action (add-not-present | add-or-replace | strip)` | Specifies the action to be performed on the parameter.
  
**Example:**  
Router(config-sbc-sbe-sip-prm-ele)# action add-not-present value phone

**Step 7**  
`end` | Exits the SBE parameter profile parameter configuration mode and returns to Privileged EXEC mode.
  
**Example:**  
Router(config-sbc-sbe-sip-prm-ele)# end

**Step 8**  
`show sbc sbc-name sbe sip-parameter-profile [profile name]` | Displays details for the parameter profile with the designated name.
  
Use the name default to view the default profile.
  
**Example:**  
Router# show sbc mysbc sbe sip parameter-profile profile1

**Step 9**  
`show sbc sbc name sbe sip essential-headers` | Displays a list of the essential headers.
  
**Example:**  
Router# show sbc mysbc sbe sip essential-headers

### Applying a Parameter Profile to a Header Profile

Perform this task to apply parameter profiles to a header profile.

### SUMMARY STEPS

1. `configure terminal`
2. `sbc sbc-name`
3. `sbe`
4. `sip header-profile header-profile-name`
5. `header header-name`
6. `parameter-profile parameter-profile-name`
7. `end`
8. `show sbc sbc-name sbe sip header-profile (profile-name)`
### DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 1**  
configure terminal                | Enters global configuration mode.                                       |
| **Example:**                             |                                                                         |
| Router# configure terminal                |                                                                         |
| **Step 2**  
sbc sbc-name                        | Enters the configuration mode of an SBC service.                       |
| **Example:**                             |                                                                         |
| Router(config)# sbc mysbc                | Use the `sbc-name` argument to define the name of the service.          |
| **Step 3**  
sbe                                | Enters the configuration mode of the signaling border element (SBE) function of the SBC. |
| **Example:**                             |                                                                         |
| Router(config-sbc)# sbe                  |                                                                         |
| **Step 4**  
sip header-profile header-profile-name | Enters the configuration mode for a header profile.                     |
| **Example:**                             |                                                                         |
| Router(config-sbc-sbe-sip)# sip header-profile profile1 |                                                                         |
| **Step 5**  
header header-name                  | Enters the header subcommand mode, where you specify the header type to match. |
| **Example:**                             |                                                                         |
| Router(config-sbc-sbe-sip-hdr)# header P-Asserted-Identity |                                                                         |
| **Step 6**  
parameter-profile parameter-profile-name | Configures the parameter profile to apply when the header type is matched. |
| **Example:**                             |                                                                         |
| Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile parmprof1 |                                                                         |
| **Step 7**  
end                                | Exits the SIP header profile header configuration mode and returns to Privileged EXEC mode. |
| **Example:**                             |                                                                         |
| Router(config-sbc-sbe-sip-hdr-ele)# end  |                                                                         |
| **Step 8**  
show sbc sbc-name sbc sip header-profile name | Displays the header profile information.                                |
| **Example:**                             |                                                                         |
| Router# show sbc sbc-name sbc sip header-profile name |                                                                         |
Associating with an Adjacency

Perform the following steps to associate a header profile with an adjacency.

**SUMMARY STEPS**

1. `configure terminal`
2. `sbc sbc-name`
3. `sbe`
4. `adjacency sip adjacency-name`
5. `header-profile inbound profile-name`
6. `end`
7. `show sbc sbc-name sbe sip header-profile name`

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><code>configure terminal</code></td>
</tr>
<tr>
<td>Example:</td>
<td>Enables the global configuration mode.</td>
</tr>
<tr>
<td></td>
<td>Router# configure terminal</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><code>sbc sbc-name</code></td>
</tr>
<tr>
<td>Example:</td>
<td>Enters the mode of an SBC service.</td>
</tr>
<tr>
<td></td>
<td>Use the <code>sbc-name</code> argument to define the name of the service.</td>
</tr>
<tr>
<td></td>
<td>Router(config)# sbc mySBC</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><code>sbe</code></td>
</tr>
<tr>
<td>Example:</td>
<td>Enters the mode of an SBE entity within an SBC service.</td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc)# sbe</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><code>adjacency sip adjacency-name</code></td>
</tr>
<tr>
<td>Example:</td>
<td>Enters the mode of an SBE SIP adjacency.</td>
</tr>
<tr>
<td></td>
<td>Use the <code>adjacency-name</code> argument to define the name of the service.</td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe)# adjacency sip sipGW</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><code>header-profile inbound profile-name</code></td>
</tr>
<tr>
<td>Example:</td>
<td>Sets profile1 to be used for inbound signaling on adjacency sipGW.</td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe-adj-sip)# header-profile inbound profile1</td>
</tr>
</tbody>
</table>
SIP Message Editing Using Profiles

Chapter 23      SIP Message Manipulation

SIP Message Editing Using Profiles

Ability to Insert Firewall Parameter in the SIP Contact Header

This feature enables Cisco Unified Border Element (SP Edition) to insert the calling party’s network information (IP address) into SIP headers.

You can use this feature to insert the public IP address for user equipment (UE) that is behind the Network Address Translation (NAT) devices into the SIP contact header as a “firewall” parameter. Inserting a firewall parameter in the header is needed because public IP address information in SIP messages is required in order to properly charge the related parties.

A sample modified contact header in SIP message is the following:

Contact: <sip:ea7cf5084c04f49e77644dbe53fd5f1d@10.140.90.6;transport=udp;firewall=10.0.48.41>; Expires=600

See ?$paranum>Ability to Insert Firewall Parameter in SIP Contact Header Examples? section on page 23-64 for examples on inserting IP address information into SIP contact headers.

Configuring Ability to Insert Firewall Parameter in the SIP Contact Header

Perform these tasks to configure this feature.

SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip parameter-profile profile-name
5. parameter {parameter name}
6. action {add-not-present [value] {private-ip-address | public-ip-address | access-user-data} |
add-or-replace [value] {private-ip-address | public-ip-address | access-user-data} | strip}
7. exit
8. sip parameter-profile profile-name
9. parameter {parameter name}
10. action {add-not-present [value] {private-ip-address | public-ip-address | access-user-data} |
add-or-replace [value] {private-ip-address | public-ip-address | access-user-data} | strip}
11. exit

Example:

Router(config-sbc-sbe-sip-hdr-prf)# end

Exits the header profile mode and returns to Privileged EXEC mode.

Example:

Router# show sbc sbc-name sbe sip header-profile name

Displays the header profile information.
12. sip header-profile profile-name
13. action {add-not-present [value] {private-ip-address | public-ip-address | access-user-data}] | add-or-replace [value] {private-ip-address | public-ip-address | access-user-data}] | strip}
14. exit
15. header header-name
16. entry entry_num {action {add-header | as-profile | drop-msg | pass | replace-name | replace-value | strip} | parameter-profile name}
17. parameter-profile name
18. sip header-profile profile-name
19. header header-name
20. entry entry_num {action {add-header | as-profile | drop-msg | pass | replace-name | replace-value | strip} | parameter-profile name}
21. parameter-profile name

**DETAILED STEPS**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong></td>
<td><strong>configure terminal</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Configures a parameter profile and enters SBE SIP header configuration mode.</td>
</tr>
<tr>
<td><strong>Step 2</strong></td>
<td><strong>sbc</strong> sbc-name</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Enters the configuration mode of the SBE function of the SBC. Use the sbc-name argument to define the name of the service.</td>
</tr>
<tr>
<td><strong>Step 3</strong></td>
<td><strong>sbe</strong></td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Enters the configuration mode of the SBE function of the SBC.</td>
</tr>
<tr>
<td><strong>Step 4</strong></td>
<td><strong>sip parameter-profile</strong> {profile-name}</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Configures a parameter profile and enters SBE SIP header configuration mode.</td>
</tr>
<tr>
<td><strong>Step 5</strong></td>
<td><strong>parameter</strong> {parameter name}</td>
</tr>
<tr>
<td><strong>Example:</strong></td>
<td>Adds a parameter with a specified name to the parameter profile and enters SIP parameter profile parameter configuration mode.</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>action {add-not-present [value] (private-ip-address</td>
<td>public-ip-address</td>
</tr>
<tr>
<td></td>
<td>add-or-replace [value] (private-ip-address</td>
<td>public-ip-address</td>
</tr>
<tr>
<td></td>
<td>strip }</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-sip-prm-ele)# action-strip</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>exit</td>
<td>Exits SBE parameter profile parameter configuration mode and enters SBE configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-sip-prm-ele)# exit</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>sip parameter-profile {profile-name}</td>
<td>Configures a parameter profile. Enters into SIP parameter profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe)# sip parameter-profile access-param</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>parameter {parameter name}</td>
<td>Adds a parameter with a specified name to the parameter profile. Enters SIP parameter profile configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-sip-prm)# parameter firewall</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>action {add-not-present [value] (private-ip-address</td>
<td>public-ip-address</td>
</tr>
<tr>
<td></td>
<td>add-or-replace [value] (private-ip-address</td>
<td>public-ip-address</td>
</tr>
<tr>
<td></td>
<td>strip }</td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-sip-hdr-ele)# action add-or-replace value public-ip-address</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>exit</td>
<td>Exits to SBE configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-sip-hdr-ele)# exit</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>sip header-profile profile-name</td>
<td>Configures a header profile. Enters SIP header profile header configuration mode.</td>
</tr>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe)# sip header-profile proxy</td>
<td>If you enter the profile-name default, the default profile is configured. This profile is used for all adjacencies which do not have a specific profile configured.</td>
</tr>
</tbody>
</table>
### Command or Action

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 13   | **action** `{add-not-present | value}
  `{private-ip-address | public-ip-address | access-user-data}` | Configures the action to take on a parameter. |
|      | add-or-replace `{value}
  `{private-ip-address | public-ip-address | access-user-data}` | |
|      | strip} | |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# action add-or-replace value public-ip-address | |
| 14   | **exit** | Exits SBE header profile header configuration mode and enters into SIP header configuration mode. |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# exit | |
| 15   | **header** name | Configures the profile to contain the header test1. Enters SBE header profile header configuration mode. |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# header test1 | |
| 16   | **entry** entry_num `{action | add-header | as-profile | drop-msg | pass | replace-name | replace-value | strip | parameter-profile name}` | Configures an entry in a profile. |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# entry 1 | |
| 17   | **parameter-profile** parameter-profile-name | Configures the parameter profile to apply when the header type is matched. |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile proxy-param | |
| 18   | **sip header-profile** profile-name | Configures a header profile. Enters SIP header configuration mode. |
| Example: | Router(config-sbc-sbe)# sip header-profile test1 | If you enter the `profile-name default`, the default profile is configured. This profile is used for all adjacencies which do not have a specific profile configured. |
| 19   | **header** name | Configures the profile to contain the header test1. Enters SBE header profile header configuration mode. |
| Example: | Router(config-sbc-sbe-sip-hdr-ele)# header test1 | |
Chapter 23      SIP Message Manipulation

SIP Message Editing Using Profiles

This section contains the following:

- Method Profile Examples, page 23-46
- Applying Method Profiles Example, page 23-48
- Associating Predefined Header Profiles Example, page 23-48
- Associating Predefined Parameter Profiles Example, page 23-49
- Associating Response Code Mapping Example, page 23-50
- Configuring Header Profiles Example, page 23-50
- Applying Header Profiles Example, page 23-51
- Header Manipulation Examples, page 23-52
- Response Filtering Example, page 23-60
- Parameter Profile Examples, page 23-61
- Ability to Insert Firewall Parameter in SIP Contact Header Examples, page 23-64

Method Profile Examples

The following example shows the commands and output generated when you configure method profiles.

Router# configure terminal
Router(config)# sbc umsbc-node3
Router(config-sbc)# sip method-profile test1
   ==> Configures new method profile with name test1
Router(config-sbc-sbe)# method abcd
   ==> Adds a method abcd to method profile test1
   by default, abcd is whitelisted if applied to the adjacency
Router(config-sbc-sbe-sip-mth)# blacklist
   ==> Blacklists abcd and allow methods other than abcd on the adjacency
Router:Nov 13 17:43:11.124: config[65761]: %MGBL-CONFIG-6-DB_COMMIT: Configuration
   committed by user 'username'. Use 'show configuration commit changes 1000000296' to view
   the changes.
Router(config-sbc-sbe-sip-mth)# end

---

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>entry entry_num</code></td>
<td>Configures an entry in a profile.</td>
</tr>
<tr>
<td><code>as-profile</code></td>
<td></td>
</tr>
<tr>
<td><code>drop-msg</code></td>
<td></td>
</tr>
<tr>
<td><code>pass</code></td>
<td></td>
</tr>
<tr>
<td><code>replace-name</code></td>
<td></td>
</tr>
<tr>
<td><code>replace-value</code></td>
<td></td>
</tr>
<tr>
<td><code>strip</code></td>
<td></td>
</tr>
<tr>
<td><code>parameter-profile name</code></td>
<td>Configures the parameter profile to apply when the header type is matched.</td>
</tr>
</tbody>
</table>

Example:

Router(config-sbc-sbe-sip-hdr-ele)# entry 1
   action as-profile

Step 21

Example:

Router(config-sbc-sbe-sip-hdr-ele)#
   parameter-profile access-param
This example shows the output for all method profiles.

This command describes the available method profiles which can be used by the adjacencies. By default, the “default” method profile is configured implicitly and applied to both inbound and outbound directions of all the adjacencies. The default method profile is always active unless it is overwritten by a user-configured method profile. “In use” explains whether the method profile is used by any adjacency or not. When the value is Yes, the “default” method profile is applied to all the adjacencies and is in use. However “test1” has been configured, but not applied to any of the adjacencies. Once you apply the test1 method profile to any adjacency, test1 shows Yes in the “In use” field.
This example shows the output for the method profiles test1.

Router# show sbc test sbe sip method-profile test1
Method profile "test1"
  Description:
  Type:  Whitelist
  Methods:
    INVITE
    action as-profile
    map-status-code
    range 50X value 500
    range 60X value 600
    Not in use with any adjacencies

Applying Method Profiles Example

The following examples show the commands and output generated when you are applying a method profile to Cisco Unified Border Element (SP Edition).

The method-profile inbound test1 command applies method profile “test1” on the inbound direction. It means that for all incoming messages, check for the method type “abcd.” If the “abcd” method arrives, blacklist it and generate error code 405 Method Not Allowed. All other methods are allowed.

Router# configure terminal
Router(config)# sbc umsbc-node3
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip sipp-10
Router(config-sbc-sbe-adj-sip)# method-profile inbound test1

Router:Nov 13 17:44:28.609 : config[65761]: %MGBL-CONFIG-6-DB_COMMIT : Configuration committed by user 'username'. Use 'show configuration commit changes 1000000297' to view the changes.
Router(config-sbc-sbe-adj-sip)# end
Router:Nov 13 17:44:31.637 : config[65761]: %MGBL-SYS-5-CONFIG_I : Configured from console by username

Router# show sbc umsbc-node3 sbe sip method-profile
Method profiles for SBC service "umsbc-node3"
Name                      In use
====================================
test1                      Yes
testb                      No

Router# show sbc umsbc-node3 sbe sip method-profile test1
Method profile "test1"
  Type:  Blacklist
  Methods:
    abcd
  In use by:
    Adjacency: sipp-10 (in)

Associating Predefined Header Profiles Example

This example shows how to ensure that the parameter myparm=myvalue is added to the request-line of an INVITE:

First, configure a parameter profile for myparm:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-profile parmprof1
Router(config-sbc-sbe-sip-prm)# parameter myparm
Router(config-sbc-sbe-sipprm-ele)# action add-not-present value myvalue

Then configure and associate with a method profile:

Router# configure terminal
Router(config)# sbe test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip method-profile mthdprof1
Router(config-sbc-sbe-sip-mth)# method INVITE
Router(config-sbc-sbe-sip-mth-ele)# parameter-profile parmprof1

Finally, associate with an adjacency

Router# configure terminal
Router(config)# sbe test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mthdprof1

At the inbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0

At the outbound side:
INVITE sip:1234567@cisco.com;user=phone;myparm=myvalue SIP/2.0

**Associating Predefined Parameter Profiles Example**

The following example shows how to ensure P-Asserted-Identity is always passed in an INVITE if it contains user=phone.

First, configure a header profile which references a P-Asserted-Identity header:

Router# configure terminal
Router(config)# sbe test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile hdrprof1
Router(config-sbc-sbe-sip-hdr)# header P-Asserted-Identity
Router(config-sbc-sbe-sip-hdr-ele)# action pass
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-value contains user=phone

Then create and associate the header profile with a method profile:

Router(config-sbc-sbe)# sip method-profile mthdprof1
Router(config-sbc-sbe-sip-mth)# method INVITE
Router(config-sbc-sbe-sip-prm-ele)# header-profile hdrprof1
Finally, associate with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mthdprof1
```

At the inbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
P-Asserted-Identity: "rob" <sip:1234567@cisco.com;user=phone>
```

At the outbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
P-Asserted-Identity: "rob" <sip:1234567@cisco.com;user=phone>
```

**Associating Response Code Mapping Example**

The following example shows how to create a status-code map so that all 5XX responses to an INVITE are mapped to 500.

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip method-profile mthdprof1
Router(config-sbc-sbe-sip-mth)# method INVITE
Router(config-sbc-sbe-sip-mth-ele)# map-status-code
Router(config-sbc-sbe-sip-mth-ele-map)# range 5XX value 500
```

Finally, associate with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# method-profile outbound mthdprof
```

At the inbound side:

```
SIP/2.0 501 Not Implemented
```

At the outbound side:

```
SIP/2.0 500 Internal Server Error
```

**Configuring Header Profiles Example**

The following example shows the commands and output generated when you configure the header profiles.

```
Router(config)# sbc umsbc-node3 sbe
Router(config-sbc-sbe)# sip header-profile EXAMPLE
Router(config-sbc-sbe-sip-hdr)# blacklist
Router(config-sbc-sbe-sip-hdr)# header abcd
Router# show sbc sbe4 sbe sip header-profile EXAMPLE
```

Header profile EXAMPLE

**Type:** Whitelist

**Headers:**

- abcd
Applying Header Profiles Example

The following example shows the commands and output generated when you are applying a header profile to Cisco Unified Border Element (SP Edition).

```
Router# configure terminal
Router(config)# sbc umsbc-node3 sbe
Router(config-sbc-sbe)# adjacency sip sipp-10
Router(config-sbc-sbe-adj-sip)# header-profile inbound test1
Router(config-sbc-sbe-adj-sip)# header-profile outbound test1
Router# show sbc umsbc-node3 sbe sip header-profile test1

Header profile "test1"
  Type:   Blacklist
  Headers:
    abcd
  In use by:
    Adjacency: sipp-10 (in, out)

show running-config

sbc umsbc-node3 sbe activate

sip header-profile test1
  blacklist
    header abcd

 adjacency sip sipp-10
  header-profile inbound test1
  header-profile outbound test1
  signaling-address ipv4 88.88.109.8
  signaling-port 5060
  remote-address ipv4 10.10.105.222 255.255.255.255
  security trusted-encrypted
  signaling-peer 10.10.105.222
  signaling-peer-port 5060
```
account sip-customer

Header Manipulation Examples

Example—Removing P-Asserted-Identity Header

The following example shows how to remove the header in any message if the header P-Asserted-Identity contains user=phone.

First, access the header:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header P-Asserted-Identity
Router(config-sbc-sbe-hdr-ele)# action strip
Router(config-sbc-sbe-hdr-ele-act)# condition header-value contains user=phone
```

Next, associate the header with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:

```
P-Asserted-Identity: "rob" <sip:1234567@cisco.com;user=phone>
```

At the outbound side:

```
No P-Asserted-Identity header present
```

Add this condition in addition to a previous existing condition:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header P-Asserted-Identity
Router(config-sbc-sbe-hdr-ele)# entry 2
Router(config-sbc-sbe-hdr-ele-act)# condition header-value contains user=phone
```

Finally, associate the header profile with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
P-Asserted-Identity: "rob" <sip:1234567@cisco.com;user=phone>
```
At the outbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...<No P-Asserted-Identity header present>
```

**Example—Removing Header Based on Condition in Another Header**

The next example shows how to remove a header based on a condition in another header in the message. First, strip the P-Asserted-Identity header, but only if Call-Info: contains "telephone-event."

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header P-Asserted-Identity
Router(config-sbc-sbe-hdr-ele)# action strip
Router(config-sbc-sbe-hdr-ele-act)# condition header-name Call-Info header-value contains telephone-event
```

Then associate the header profile with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...P-Asserted-Identity: "rob" <sip:1234567@cisco.com;user=phone>
...Call-Info: <sip:8985010.131.132.6>;method="NOTIFY;Event=telephone-event;Duration=1000"
```

The result at the outbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...<No P-Asserted-Identity header present>
```

**Example—Removing Organization Header from All Responses**

The next example removes an Organization header from all Responses:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Organization
Router(config-sbc-sbe-hdr-ele)# action strip
Router(config-sbc-sbe-hdr-ele-act)# condition status-code eq 200
```

Associate the header profile with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```
At the inbound side:
SIP/2.0 200 OK
...
Allow: INVITE, ACK, PRACK, SUBSCRIBE, BYE, CANCEL, NOTIFY, INFO, REFER, UPDATE

At the outbound side:
SIP/2.0 200 OK
...
<No allow header present>

Example—Transforming a Header into Another Header

This example transforms one header into another header (Diversion into Hist-Info).

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Diversion
Router(config-sbc-sbe-hdr-ele)# action replace-name value Hist-Info

Associate the header profile with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

At the inbound side:
INVITE sip:1234567@cisco,com;user=phone SIP/2.0
...
Diversion: <sip:1234567@cisco.com>;reason=unconditional;counter=1;privacy=off

At the outbound side:
INVITE sip:1234567@cisco,com;user=phone SIP/2.0
...
Hist-Info: <sip:1234567@cisco.com>;reason=unconditional;counter=1;privacy=off

Example—Outgoing Messages Contain a Specific Header

This example ensures all outgoing messages contain a specific header (Organization: Cisco.com).

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Organization
Router(config-sbc-sbe-hdr-ele)# action add-first-header value cisco.com

Associate the header profile with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Example—Blacklisting a Header

This example blacklists a header (all instances are removed for any method/response).

⚠️ Note This can only be performed against a header profile type of blacklist

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr-ele)# blacklist
Router(config-sbc-sbe-sip-hdr)# header Organization

Or:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe-hdr)# sip header-profile headprof1
Router(config-sbc-sbe-hdr-ele)# blacklist
Router(config-sbc-sbe-sip-hdr)# header Organization
Router(config-sbc-sbe-sip-hdr)# action as-profile

Associate the header profile with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

At the inbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... Organization: cisco.com

At the outbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... <no Organization header present>
Example—Whitelisting a Header

This example whitelists a header (pass in all methods/responses).

Note: This can only be specified against a whitelist type of profile which is a default profile and same as “no blacklist.”

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Organization
Or:
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Organization
Router(config-sbc-sbe-hdr-ele)# action as-profile

Associate the header profile with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

At the inbound side:

INVITE sip:1234567@cisco,com;user=phone SIP/2.0
...
Organization: cisco.com

At the outbound side:

INVITE sip:1234567@cisco,com;user=phone SIP/2.0
...
Organization: cisco.com

Example—Passing a Date Header

This example passes a header (Date) conditionally in a 200 response.

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-hdr)# header Date
Router(config-sbc-sbe-hdr-ele)# action pass
Router(config-sbc-sbe-hdr-ele-act)# condition status-code eq 200

Associate with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
At the inbound side:

Ensure no other responses contain a Date: header
SIP/2.0 200 OK
...  
Date: Mon, 01 Jan 2008 GMT

At the outbound side:

SIP/2.0 200 OK
...  
Date: Mon, 01 Jan 2008 GMT

Also try all responses containing a Date: header and ensure the 200 OK only contains one

Example—Stripping Organization Headers in INVITE

This example strips all 'Organization' headers in an INVITE. To do this, a header profile is created and then associated with it with a method profile.

Note: Header profiles can be associated with vital (essential) methods.

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headerprof1
Router(config-sbc-sbe-hdr)# blacklist
Router(config-sbc-sbe-hdr-ele)# header Organization

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip method-profile methodprof1
Router(config-sbc-sbe-sip-mth)# blacklist
Router(config-sbc-sbe-sip-mth)# method INVITE
Router(config-sbc-sbe-sip-mth-ele)# header-profile headerprof1

Associate with an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip-mth)# method-profile outbound methodprof1

At the inbound side:

INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...  
Organization: cisco.com

At the outbound side:

INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...  
<no Organization: header present>
Example—Applying Parameter Profile

This example applies a parameter profile to add user=phone into the request-line of an INVITE.

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-profile test
Router(config-sbc-sbe-sip-prm)# parameter user
Router(config-sbc-sbe-sip-prm-ele)# action add-not-present value phone
```

Associate with a method profile:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe) sip method-profile test
Router(config-sbc-sbe-sip-mth)# method INVITE
Router(config-sbc-sbe-sip-mth-ele) parameter-profile test
```

Associate with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# method-profile inbound headprof1
```

At the inbound side:

```
INVITE sip:1234567@cisco.com SIP/2.0
```

At the outbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
```

Example—Stripping P-Called-Party-Identity

This example shows how to strip the P-Called-Party-Identity and modify the To: header based on its content:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# description "store the P-Called-Party-Identity"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-name P-Called-Party-Identity
Router(config-sbc-sbe-sip-hdr-ele-act)# condition header-value store-as pcpid
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# header P-Called-Party-Identity entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# header To entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action replace-value value "$(pcpid)"
Router(config-sbc-sbe-sip-hdr-ele-act)# description "replace the To value"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable pcpid is-defined eq true
```
Associate with an outbound adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

Replacing Outbound Request Line Example

This example shows how to replace the outbound request-line with host 172.1.1.1 if user = begins with 1234:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# condition request-uri is-sip-uri eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and request-uri sip-uri-user store-as user
Router(config-sbc-sbe-sip-hdr-ele-act)# request-line entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action replace-value value "sip:${user}@172.1.1.1"
Router(config-sbc-sbe-sip-hdr-ele-act)# description "convert RPID param into Privacy header value"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and request-uri sip-uri-user regex-match "^1234"

Associate with an outbound adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

Example—P-KT-UE-IP Header Support

The P-KT-UE-IP header is a type of private header that is supported as a type of SIP header manipulation. The examples in this section show how to remove any existing P-KT-UE-IP headers from all received messages and then replace them with a single P-KT-UE-IP header for INVITE and OOD requests. In the examples, the call is placed from adj1 to adj2.

The following shows how to configure a header profile with two entries. The first entry strips the "P-KT-UE-IP" header and the second entry adds the "P-KT-UE-IP" with a value set to the 18-character string ${msg.rmt_ip_addr}.

Router(config-sbc-sbe)# sip header-profile kt
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# condition adjacency signaling-peer store-as address
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# header P-KT-UE-IP
Router(config-sbc-sbe-sip-hdr-ele-act)# entry 1 action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# entry 2 action add-header value "${address}"
The following applies the above header profile to the incoming adjacency as an inbound header profile.

Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# header-profile inbound kt

The following configures a header profile to allow passthrough of the "P-KT-UE-IP" header.

Router(config-sbc-sbe)# sip header-profile kt-pass
Router(config-sbc-sbe-sip-hdr)# header P-KT-UE-IP
Router(config-sbc-sbe-sip-hdr-ele)# action pass

The following applies the above header profile to the outgoing adjacency as an outbound header profile.

Router(config-sbc-sbe)# adjacency sip adj2
Router(config-sbc-sbe-adj-sip)# header-profile outbound kt-pass

Response Filtering Example

The following example drops SIP 183 provisional responses from a header profile based on matching the header * associated with inbound and outbound adjacencies.

First, create a header profile headprof1 to match on header * and drop the message:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# header *
Router(config-sbc-sbe-sip-hdr-ele)# action drop-msg
Router(config-sbc-sbe-sip-hdr-ele-act)# condition status-code eq 183

Associate the profile headprof1 to the inbound side of an adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adjacencyA
Router(config-sbc-sbe-adj-sip)# header-profile inbound headerprof1

Associate the profile headprof1 to the inbound and outbound sides of another adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adjacencyB
Router(config-sbc-sbe-adj-sip)# header-profile inbound headerprof1
Router(config-sbc-sbe-adj-sip)# header-profile outbound headerprof1
Parameter Profile Examples

This example shows how to add a user=phone parameter into the To: header if one has not already been specified in a header.

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-profile parmprof1
Router(config-sbc-sbe-sip-prm)# parameter user
Router(config-sbc-sbe-sip-prm-ele)# action add-not-present value phone
```

Now add to a header profile:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# header To
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile parmprof1
```

Now associate with an adjacency:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com>;tag=1234;
```

At the outbound side:

```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com;user=phone>;tag=1234
```

This example removes the 'user' parameter ('user=phone','user=fax' ...) from the To: header.

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe-sbe)# sip parameter-profile parmprof1
Router(config-sbc-sbe-sip-prm)# parameter user
Router(config-sbc-sbe-sip-prm-ele)# action strip
```

Add to a header profile:

```
Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# header To
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile parmprof1
```
Finally, associate with an adjacency:

```
Router# configure terminal
Router(config)# sbe test
Router(config-sbe)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:
```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com;user=phone;tag=1234;
```

At the outbound side:
```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com>;tag=1234
```

This example shows how to replace 'user=phone' parameter with user=fax or to add user=fax if a user parameter is not present in the header.

```
Router# configure terminal
Router(config)# sbe test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-profile parmprof1
Router(config-sbc-sbe-sip-prm)# parameter user
Router(config-sbc-sbe-sip-prm-ele)# action add-or-replace value fax
```

Add to a header profile:
```
Router# configure terminal
Router(config)# sbe test
Router(config-sbe)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# header To
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile parmprof1
```

Finally, associate with an adjacency:
```
Router# configure terminal
Router(config)# sbe test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1
```

At the inbound side:
```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com;user=phone;tag=1234;
```

At the outbound side:
```
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
...
To: "rob" <sip:1234567@cisco.com;user=fax>;tag=1234
```

Or
At the inbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com;tag=1234;

At the outbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com;user=fax;tag=1234

The next example adds 'user=phone' parameter if it is not already present in the header.

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-profile parmprof1
Router(config-sbc-sbe-sip-prm)# parameter user
Router(config-sbc-sbe-sip-prm-ele)# action add-not-present value phone

Add parameter profile to a header profile:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-profile headprof1
Router(config-sbc-sbe-sip-hdr)# header To
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile parmprof1

Finally, associate with an adjacency

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-profile outbound headprof1

At the inbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com;user=fax;tag=1234;

At the outbound side:
No parameter added as a user parameter already exists
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com>;tag=1234

Or

At the inbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com;tag=1234;

At the outbound side:
INVITE sip:1234567@cisco.com;user=phone SIP/2.0
... 
To: 'rob' <sip:1234567@cisco.com;user=phone>;tag=1234
Chapter 23      SIP Message Manipulation

SIP Message Editing Using Editors

Ability to Insert Firewall Parameter in SIP Contact Header Examples

This example adds a SIP parameter profile to remove or append the parameter called firewall:

Router(config-sbc-sbe)# sip parameter-profile proxy-param
Router(config-sbc-sbe-sip-prm)# parameter firewall
Router(config-sbc-sbe-sip-prm-ele)# action strip
Router(config-sbc-sbe-sip-prm-ele)# sip parameter-profile access-param
Router(config-sbc-sbe-sip-prm)# parameter firewall
Router(config-sbc-sbe-sip-prm-ele)# action add-or-replace value public-ip-address

This example adds a SIP header profile and associates the parameter profile with the header profile:

Router(config-sbc-sbe-sip-prm-ele)# sip header-profile proxy
Router(config-sbc-sbe-sip-hdr)# header contact entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action as-profile
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile proxy-param
Router(config-sbc-sbe-sip-hdr-ele)# sip header-profile access
Router(config-sbc-sbe-sip-hdr)# header contact
Router(config-sbc-sbe-sip-hdr-ele)# entry 1 action as-profile
Router(config-sbc-sbe-sip-hdr-ele)# parameter-profile access-param

This example adds a SIP header profile to a SIP adjacency:

adjacency sip sip-proxy
  header-profile inbound proxy
  header-profile outbound access
  adjacency sip sip-user
  header-profile inbound access
  header-profile outbound proxy

SIP Message Editing Using Editors

Note

This section describes body, header, method, option, and parameter editors. The ?Sparanum>SDP Editing Using Script-Based Editors? section on page 23-84 describes script-based editors for modifying the SDP content in SIP messages. You can apply any combination of both types of editors on the SBC for editing SIP messages.

In Release 2.4S, profiles were introduced to enable the SBC to conditionally modify SIP messages. You could configure a profile to modify the body, header, method, option, or parameter of SIP messages that met the matching criteria you specified. This approach was flexible but posed the following limitations:

- Matching criteria could not be set for the vital parts of a message because there was a probability of the call failing if the vital parts of the message were modified.
- With certain limited exceptions, the vital parts of a message could not be modified because the original content of these vital parts was not available at the point at which the profiles were applied.

From Release 3.3S, the concept of editors has been introduced. An editor refers to any kind of SBC configuration that is used for conditionally editing SIP messages. Profiles that were introduced in earlier releases are now renamed as editors. For example, body profiles are now known as body editors, header profiles are known as header editors, and so on.

Editors can be associated with an adjacency and linked together so that they can be applied in a specified sequence at run time. In addition, you can test editors by applying them on a test message (a SIP INVITE). You can use the output of the test to determine whether the editors meet your requirements.
In Cisco IOS XE Release 3.3S, the following additional enhancements have been introduced in the SIP Message Editing feature:

- **To and From multimode fiber optic edits**

  Prior to Cisco IOS XE Release 3.3S, the To and From outbound headers of only out-of-dialog messages and dialog-creating messages could be edited. After an edit was performed on a dialog-creating message, the edit was automatically propagated across all the new messages sent on the dialog. From Cisco IOS XE Release 3.3S, edits on the To and From headers can also be performed on in-dialog messages. There is no automatic propagation of these edits. This requires you to ensure that the edits are consistently performed for all messages sent on the dialog.

- **Resource Priority header inspection**

  Prior to Cisco IOS XE Release 3.3S, the Resource Priority header inspection function examined a message before any inbound MMF editing was performed. From Cisco IOS XE Release 3.3S, the Resource Priority header inspection function examines a message after inbound editing has been performed.

- **100rel_required match condition variable**

  Prior to Cisco IOS XE Release 3.3S, the 100rel_required match condition variable was a call property that was updated when new information about 100rel support came in from each call leg. From Cisco IOS XE Release 3.3S, this variable is an indicator of whether the received message is marked as Required: 100rel.

- **Failure responses**

  Prior to Cisco IOS XE Release 3.3S, failures encountered during message editing resulted in the SBC sending a rejection for the unedited message. From Cisco IOS XE Release 3.3S, the response contains the state of the message at the point of failure. For example, headers added during editing are mentioned in the failure response.

The following sections provide information about implementing SIP message editing using body, header, method, option, and parameter editors:

- **Restrictions for SIP Message Editing, page 23-65**
- **Guidelines for Naming Editors, page 23-66**
- **Configuring Editors, page 23-66**
- **Configuration Examples for SIP Message Editors, page 23-76**

## Restrictions for SIP Message Editing

The SIP Message Editing feature does not support the following actions:

- Editing To and From header tags
- Applying the pass and strip actions on To and From header tags
- Outbound editing of Via headers
- Changing the method types of INVITE, CANCEL, and ACK messages
Guidelines for Naming Editors

Apply the following guidelines while naming an editor:

- Ensure that each editor has a unique name. Apply this guideline across editors. For example, ensure that the name of a header editor is not the same as the name of a method editor.
- Note that an editor and a profile should have the same name to ensure an easy migration path.

Configuring Editors

This task describes how to configure editors on the SBC.

SUMMARY STEPS

1. configure terminal
2. sbc sbc-name
3. sbe
4. sip editor-type {editor | profile}
5. sip body-editor editor-name
6. exit
7. sip method-editor {editor-name | default}
8. exit
9. sip option-editor {editor-name | default}
10. exit
11. sip parameter-editor {editor-name | default}
12. exit
13. sip header-editor {editor-name | default}
14. exit
15. adjacency sip adjacency-name
16. editor-type {editor | profile}
17. header-editor {inbound | outbound} {editor-name | default}
18. method-editor {inbound | outbound} {editor-name | default}
19. option-editor [ua | proxy] {inbound | outbound} {editor-name | default}
20. body-editor {inbound | outbound} {editor-name}
21. editor-list {after-send | before-receive}
22. `editor order-number editor-name [condition [body contains sdp]]`
23. `end`
24. `show sbc sbc-name sbe editors`
25. `show sbc sbc-name sbe sip header-editor [editor-name]`
26. `show sbc sbc-name sbe sip body-editor [editor-name]`
27. `show sbc sbc-name sbe sip method-editor [editor-name]`
28. `show sbc sbc-name sbe sip option-editor [editor-name]`
29. `show sbc sbc-name sbe sip parameter-editor [editor-name]`
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 1</strong> configure terminal</td>
<td>Enters the global configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td><strong>Step 2</strong> sbc sbc-name</td>
<td>Enters the SBC service mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config)# sbc mysbc</td>
<td><em>sbc-name</em>—Name of the SBC.</td>
</tr>
<tr>
<td><strong>Step 3</strong> sbe</td>
<td>Enters the SBE configuration mode of the SBC.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td><strong>Step 4</strong> sip editor-type {editor</td>
<td>profile}</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe)# sip editor-type editor</td>
<td><em>editor</em>—Sets the default for using the method, header, option, parameter, or body editor.</td>
</tr>
<tr>
<td><strong>Step 5</strong> sip body-editor editor-name</td>
<td>Creates a body editor to filter non-SDP message bodies from incoming and outgoing SIP messages.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe)# sip body-editor BodyEditor1</td>
<td><em>editor-name</em>—Specifies the name of the body editor.</td>
</tr>
<tr>
<td><strong>Step 6</strong> exit</td>
<td>Exits the SIP body configuration mode and enters the SBE configuration mode.</td>
</tr>
<tr>
<td><strong>Example:</strong> Router(config-sbc-sbe-mep-bdy)# exit</td>
<td></td>
</tr>
</tbody>
</table>

---

Note: The steps provided are for configuring SIP message manipulation using editors in a Cisco Unified Border Element (SBE). The configuration involves entering various modes and using specific commands to define how SIP messages are handled. The commands and contexts (e.g., configure terminal) are essential for navigating the device configuration and setting up the desired functionalities.
### Chapter 23  
SIP Message Manipulation

#### SIP Message Editing Using Editors

<table>
<thead>
<tr>
<th>Step</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| 7    | `sip method-editor {editor-name | default}` | Configures a method editor.  
  - *editor-name*—Specifies the name of the method editor.  
  - *default*—Configures the default method editor. This editor is used for all the adjacencies that do not have a specific editor configured.  
Example:  
Router(config-sbc-sbe)# sip method-editor MethodEditor1 |
| 8    | `exit`            | Exits the SIP method configuration mode and enters the SBE configuration mode.  
Example:  
Router(config-sbc-sbe-mep-mth)# exit |
| 9    | `sip option-editor {editor-name | default}` | Configures an option editor.  
  - *editor-name*—Specifies the name of the option editor.  
  - *default*—Configures the default option editor.  
Example:  
Router(config-sbc-sbe)# sip option-editor OptionEditor1 |
| 10   | `exit`            | Exits the SIP option configuration mode and enters the SBE configuration mode.  
Example:  
Router(config-sbc-sbe-mep-opt)# exit |

- *blacklist*—Sets this editor to be blacklist.  
- *description*—Sets the description for this editor.  
- *method*—Adds a method to this editor.  
- *map-status-code*—Allows mapping of the response codes received for a method.  

The `method` command enters the SIP method editor element configuration mode, where the following commands can be used:  

- *action*—Specifies the action performed on the method.  
- *body-editor*—Adds a body editor to act on the method.  
- *header-editor*—Adds a header editor to act on the method.  
- *map-status-code*—Allows mapping of the response codes received for a method.  

The `option` command enters the SIP option configuration mode, where the following commands can be used:  

- *action*—Specifies the action performed on the option.  
- *body-editor*—Adds a body editor to act on the option.  
- *header-editor*—Adds a header editor to act on the option.  
- *map-status-code*—Allows mapping of the response codes received for a method.  

- *blacklist*—Sets this editor to be blacklist.  
- *description*—Sets the description for this editor.  
- *option*—Adds an option to this editor.
### Command or Action

<table>
<thead>
<tr>
<th>Step 11</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| sip parameter-editor editor-name | **Example:**
Router(config-sbc-sbe)# sip parameter-editor ParameterEditor1 | Configures a parameter editor.
- *editor-name*—Specifies the name of the parameter editor.
Enters the SIP parameter configuration mode. Use the following commands under this mode to configure the parameter editor:
- **blacklist**—Sets this editor to be blacklist.
- **description**—Sets the description for this editor.
- **parameter**—Adds an parameter to this editor.
The **parameter** command enters the SIP parameter editor element configuration mode, from where you can configure the action to be taken on an element type in the parameter editor using the **action** command. |

<table>
<thead>
<tr>
<th>Step 12</th>
<th>Command</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| exit | **Example:**
Router(config-sbc-sbe-mep-prm)# exit | Exits the SIP parameter configuration mode and enters the SBE configuration mode. |
### Command or Action

**Step 13**

```
sip header-editor {editor-name | default}
```

**Example:**

```
Router(config-sbc-sbe)# sip header-editor HeaderEditor1
```

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>`sip header-editor {editor-name</td>
<td>default}`</td>
</tr>
<tr>
<td>- <code>editor-name</code></td>
<td>Specifies the name of the header editor.</td>
</tr>
<tr>
<td>- <code>default</code></td>
<td>Configures the default header editor.</td>
</tr>
</tbody>
</table>

Enters the SIP header configuration mode. Use the following commands under this mode to configure the header editor:

- `blacklist`—Sets this editor to be blacklist.
- `description`—Sets the description for this editor.
- `div-address`—Specifies a priority list of headers from which the diverted-by number is to be derived (inbound only). Enters the SIP header editor diversion header configuration mode, from where you can use the following command:
  - `header-prio`—Specifies a priority-ordered list for extracting the diverted-by address.
- `dst-address`—Specifies a priority list of headers from which the called party address is to be derived (inbound only). Enters the SIP header editor destination header configuration mode, from where you can use the following command:
  - `header-prio`—Specifies a priority ordered list for extracting the destination address.
- `header`—Adds a header to this editor. Enters the SIP header editor header configuration mode, from where you can use the following commands:
  - `action`—Specifies the type of action. Enters the SIP header editor header action mode, from where you can use the `condition` command to specify one or more conditions for the action to be effective and the `parameter-editor` command to specify the parameter editor.
  - `parameter-editor`—Specifies the parameter editor.
SIP Message Editing Using Editors

### Chapter 23      SIP Message Manipulation

#### SIP Message Editing Using Editors

**Command or Action** | **Purpose**
--- | ---
| request-line—Allow actions to modify the Request Line (outbound side only). Enters the SIP header editor header configuration mode, from where you can use the following commands:
  - action—Specifies the type of action. Enters the SIP header editor header action mode, from where you can use the condition command to specify one or more conditions for the action to be effective and the parameter-editor command to specify the parameter editor.
  - parameter-editor—Specifies the parameter editor.
| src-address—Specifies a priority list of headers from which the calling party address is to be derived (inbound only). Enters the SIP header editor calling party configuration mode, from where you can use the following command:
  - header-prio—Specifies a priority ordered list for extracting the source address.
| store-rule—Creates a store rule to extract variables from headers. Enters the SIP header editor header action configuration mode, from where you can use the following commands:
  - condition—Specifies one or more conditions for the action to be effective.
  - description—Sets the description for this action.

**Step 14** exit

**Example:**

```
Router(config-sbc-sbe-mep-hdr)# exit
```

Exits the SIP header configuration mode and enters the SBE configuration mode.

**Step 15** adjacency sip adjacency-name

**Example:**

```
Router(config-sbc-sbe)# adjacency sip SIPP
```

Enters the SBE SIP adjacency configuration mode.

- adjacency-name—Name of the service.

**Step 16** editor-type {editor | profile}

**Example:**

```
Router(config-sbc-sbe-sip)# editor-type editor
```

Specifies the editor type for the SIP adjacency to apply.

- editor—Uses the method, header, option, parameter, or body editor.
- profile—Uses the method, header, option, parameter, or body profile.
## Chapter 23  SIP Message Manipulation

### SIP Message Editing Using Editors

| Step 17 | **header-editor** (inbound | outbound) (editor-name | default) |
|---------|-----------------------------|-------------------------|
| Example: | Router(config-sbc-sbe-sip)# header-editor inbound HeaderEditor1 |

Sets a specified header editor for inbound and outbound signaling on the SBE SIP adjacency.
- **inbound**—Sets the inbound SIP header editor.
- **outbound**—Sets the outbound SIP header editor.
- **editor-name**—Name of the header editor to be set for inbound or outbound signaling on the adjacency.
- **default**—Sets the header editor to the default settings.

| Step 18 | **method-editor** (inbound | outbound) (editor-name | default) |
|---------|-----------------------------|-------------------------|
| Example: | Router(config-sbc-sbe-sip)# method-editor inbound HeaderEditor1 |

Configures the method editor.
- **inbound**—Sets the inbound SIP method editor.
- **outbound**—Sets the outbound SIP method editor.
- **editor-name**—Name of the method editor to be set for inbound or outbound signaling on the adjacency.
- **default**—Sets the method editor to the default settings.

| Step 19 | option-editor [ua | proxy] (inbound | outbound) (editor-name | default) |
|---------|-----------------------------|-------------------------|
| Example: | Router(config-sbc-sbe-adj-sip)# option-editor ua inbound OptionHeader1 |

Sets the adjacency to use the specified editor for white or blacklisting options.
- **ua**—Sets the SIP ua option editors.
- **proxy**—Sets the SIP proxy option editors.
- **inbound**—Sets the inbound SIP option editors.
- **outbound**—Sets the outbound SIP option editors.
- **editor-name**—Name of editor to use.
- **default**—Sets the method editor to the default settings.

| Step 20 | **body-editor** (inbound | outbound) (editor-name) |
|---------|-----------------------------|
| Example: | Router(config-sbc-sbe-adj-sip)# body-editor inbound BodyEditor1 |

Associates a body editor to the SIP adjacency so that the body editor acts on incoming and outgoing SIP messages.
- **inbound**—Associates the body editor to act on inbound messages on the SIP adjacency.
- **outbound**—Associates the body editor to act on outbound messages on the SIP adjacency.

**Note** When the message is passed through the SBC, the body editor is applied in both the inbound and outbound directions on the respective adjacencies on which the message is routed.
- **editor-name**—Specifies a name for the body editor. The maximum length is 30 characters.
## Command or Action

### Step 21
```
editor-list (after-send | before-receive)
```

**Example:**
```
Router(config-sbc-sbe-adj-sip)# editor-list
after-send
```

**Purpose:**
Configures a list of registered editors.
- **after-send**—Specifies that the outgoing message must be edited after it is processed by the adjacency and just before it is forwarded from the adjacency.
- **before-receive**—Specifies that the incoming message must be edited just after it is received on the adjacency and before the adjacency begins processing it.

### Step 22
```
editor order-number editor-name [condition [body contains sdp]]
```

**Example:**
```
Router(config-sbc-sbe-adj-sip-ed)# editor 1
bodyeditor1
```

**Purpose:**
Configures an editor in the editor list. For each editor that you want to apply in a sequence, run this command to specify the order of the editor in the editor list.

**Note**
You can add any combination of script-based editors and body, header, method, option, and parameter editors in the editor list.

- **order-number**—Order in which the editor must be applied. The range is from 1 to 2147483647.
- **editor-name**—Specifies the name of the editor that you want to apply to messages that are processed by the adjacency.
- **condition**—Specifies that there are one or more conditions for the editor to be applied.
- **body contains sdp**—Specifies that the message body must be SDP-based content. The editor is applied only if this condition is met. Include **body contains sdp** in the command for script-based editors.

### Step 23
```
end
```

**Example:**
```
Router(config-sbc-sbe-adj-sip)# end
```

**Purpose:**
Exits the SIP editor configuration mode, and enters the privileged EXEC mode.

### Step 24
```
show sbc sbc-name sbe editors
```

**Example:**
```
Router# show sbc mysbc sbe editors
```

**Purpose:**
Lists all the configured editors.
<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
</table>
| **Step 25**  
show sbc sbc-name sbe sip body-editor [editor-name]                          | Displays the details of all body editors, or displays details pertaining to the specified body editor. |
| **Example:**  
Router# show sbc mysbc sbe sip body-editor                                     | BodyEditor1                                                                                 |
| **Step 26**  
show sbc sbc-name sbe sip header-editor [editor-name]                           | Displays the details of all header editors, or displays details pertaining to the specified header editor. |
| **Example:**  
Router# show sbc mysbc sbe sip header-editor                                      | HeaderEditor1                                                                               |
| **Step 27**  
show sbc sbc-name sbe sip method-editor [editor-name]                            | Displays the details of all method editors, or displays details pertaining to the specified method editor. |
| **Example:**  
Router# show sbc mysbc sbe sip method-editor                                       | MethodEditor1                                                                               |
| **Step 28**  
show sbc sbc-name sbe sip option-editor [editor-name]                             | Displays the details of all option editors, or displays details pertaining to the specified option editor. |
| **Example:**  
Router# show sbc mysbc sbe sip option-editor                                       | OptionEditor1                                                                              |
| **Step 29**  
show sbc sbc-name sbe sip parameter-editor [editor-name]                           | Displays the details of all parameter editors, or displays details pertaining to the specified parameter editor. |
| **Example:**  
Router# show sbc mysbc sbe sip parameter-editor                                     | ParameterEditor1                                                                           |
Configuration Examples for SIP Message Editors

This section contains the following examples:

- Method Editor Example, page 23-76
- Header Editor Example, page 23-78
- Body Editor Example, page 23-81
- Option Editor Example, page 23-83
- Parameter Editor Example, page 23-83

Method Editor Example

The following example shows how to configure the test1 method editor and the abcd method type on the SBC2 SBC.

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip method-editor test1
Router(config-sbc-sbe-mep-mth)# method abcd
Router(config-sbc-sbe-mep-mth)# blacklist

The following example shows how the `show sbc sbe sip method-editor` command is used to display details of the meditor1 method editor and the test1 method editor before they have been applied to an adjacency.

Router# show sbc SBC2 sbe sip method-editor meditor1
method-editor "meditor1"
  Description:
  Type:    Whitelist
  Methods:
     INVITE
        action as-editor
     map-status-code
        range 5XX value 500
        range 6XX value 600
     Not in use with any adjacencies

Router# show sbc SBC2 sbe sip method-editor test1
method-editor "test1"
  Description:
  Type:    Blacklist
  Methods:
     abcd
        action as-editor
     Not in use with any adjacencies

The following example shows how the `show sbc sbe sip method-editor` command is used to display a list of all configured method editors:

Router# show sbc SBC2 sbe sip method-editor
method-editors for SBC service "SBC2"
  Name       In use
  ===============
test1        No
meditor1     No
preset-acc-in-mth No
Example—Applying the Method Editor

The `method-editor inbound test1` command applies the test1 method editor on the inbound direction. Therefore, for all incoming messages, the method type abcd is checked. When the abcd method arrives, it is blacklisted and the error code `405 Method Not Allowed` is generated. All the other methods are allowed.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbe SBC2
Router(config-sbe)# adjacency sip trans-uac
Router(config-sbe-adj-sip)# no attach
Router(config-sbe-adj-sip)# method-editor inbound test1
Router(config-sbe-adj-sip)# attach
```

The following example shows how the `show sbe sip method-editor` command is used to display details of the test1 method editor after it has been applied to an adjacency.

```
Router# show sbe sip method-editor
method-editors for SBC service "SBC2"

<table>
<thead>
<tr>
<th>Name</th>
<th>In use</th>
</tr>
</thead>
<tbody>
<tr>
<td>test1</td>
<td>Yes</td>
</tr>
<tr>
<td>meditor1</td>
<td>No</td>
</tr>
</tbody>
</table>

Router# show sbe sip method-editor test1
method-editor "test1"
  Description:
  Type: Blacklist
  Methods:
   abcd
   action as-editor
  In use by adjacency:trans-uac (in)
Header Editor Example

This section contains the following examples:

- **Example—Configuring and Applying the Header Editor**, page 23-78
- **Example—Using Directory Number Prefix to Set Privacy**, page 23-79
- **Example—Converting Remote-Party-ID or P-Preferred-Identity**, page 23-80

**Example—Configuring and Applying the Header Editor**

The following example shows how to configure the EXAMPLE header editor and the abcd header type on the SBC2 SBC.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-editor EXAMPLE
Router(config-sbc-sbe-mep-hdr)# blacklist
Router(config-sbc-sbe-mep-hdr)# header abcd
```

The following example shows how the `show sbc sbe sip header-editor` command is used to display details of the EXAMPLE header editor:

```
Router# show sbc SBC2 sbe sip header-editor EXAMPLE
header-editor "EXAMPLE"
  Description:
    Type:        Blacklist
    store-rules:
      No store-rule entries found.
    request-line:
      No request-line entries found.
    headers:
      abcd
        entry 1
          description:
            action as-editor
          Not in use with any adjacencies
          Not in use with any method-editor
```

The **header-editor inbound EXAMPLE** command and the **header-editor outbound EXAMPLE** command applies the EXAMPLE header editor on the inbound and outbound direction. Therefore, for all incoming and outgoing messages, the header type abcd is checked. When the abcd header arrives or leaves, it is blacklisted and the error code 405 *Method Not Allowed* is generated. All the other headers are allowed.

```
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip trans-uac
Router(config-sbc-sbe-adj-sip)# no attach
Router(config-sbc-sbe-adj-sip)# header-editor inbound EXAMPLE
Router(config-sbc-sbe-adj-sip)# header-editor outbound EXAMPLE
Router(config-sbc-sbe-adj-sip)# attach
```
The following example shows how the `show sbc sbe sip header-editor` command is used to display details of the EXAMPLE header editor after it has been applied to an adjacency.

Router# `show sbc SBC2 sbe sip header-editor EXAMPLE`

header-editor "EXAMPLE"
Description:
Type: Blacklist
store-rules:
  No store-rule entries found.
request-line:
  No request-line entries found.
handlers:
  abcd
    entry 1
description:
  action as-editor
In use by adjacency: trans-uac (in, out)
Not in use with any method-editor

Example—Using Directory Number Prefix to Set Privacy

This example shows how to use a directory number prefix to set privacy:

Router# `configure terminal`
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# `sbc test`
Router(config-sbc)# `sbe`
Router(config-sbc-sbe)# `sip header-editor headprof1`
Router(config-sbc-sbe-sip-hdr)# `store-rule entry 1`
Router(config-sbc-sbe-sip-hdr-ele-act)# `description "store the called party number from To"
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition is-request eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and header-name To is-sip-uri eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and header-name To sip-uri-user store-as called-dn`
Router(config-sbc-sbe-sip-hdr-ele-act)# `exit`
Router(config-sbc-sbe-sip-hdr)# `store-rule entry 2`
Router(config-sbc-sbe-sip-hdr-ele-act)# `description "store the called party number from To"`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition is-request eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and header-name To is-sip-uri eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and header-name To sip-uri-user store-as called-dn`
Router(config-sbc-sbe-sip-hdr-ele-act)# `exit`
Router(config-sbc-sbe-sip-hdr-ele-act)# `description "set $privacy based on DN"
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition variable privacy is-defined eq false`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and variable called_dn is-defined eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and variable called_dn regex-match ^\d+$`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and “none” store-as privacy`
Router(config-sbc-sbe-sip-hdr-ele-act)# `exit`
Router(config-sbc-sbe-sip-hdr-ele-act)# `store-rule entry 4`
Router(config-sbc-sbe-sip-hdr-ele-act)# `description "set $privacy based on DN"
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition variable privacy is-defined eq false`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and variable called_dn is-defined eq true`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and variable called_dn regex-match ^\d+$`
Router(config-sbc-sbe-sip-hdr-ele-act)# `condition and “user” store-as privacy`
Example——Converting Remote-Party-ID or P-Preferred-Identity

This example converts Remote-Party-ID or From into P-Preferred-Identity. If the message is a request and Remote-Party-ID is present then it stores the username into a variable username. If the From header contains a sip: URI or Tel: URI, and Remote-Part-ID was not present then it stores the username into the variable username. Strips all P-Preferred-Identity, Remote-Party-ID’s and P-Preferred-Identity headers and inserts a single P-Preferred-Identity header containing the stored username and a Privacy header based on info received:

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc)# adjacency sip adj1
Router(config-sbc-sbe)# header-editor inbound headprof1

Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# header Privacy entry 1
Router(config-sbc-sbe-sip-hdr-ele)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele)# exit
Router(config-sbc-sbe-sip-hdr)# header Privacy entry 2
Router(config-sbc-sbe-sip-hdr-ele)# action add-first-header value "${privacy}"
Router(config-sbc-sbe-sip-hdr-ele-act)# description "create a privacy header if we have privacy info"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable privacy is-defined eq true

Associate with an inbound adjacency:

Router# configure terminal
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc)# adjacency sip adj1
Router(config-sbc-sbe)# header-editor inbound headprof1

Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# store-rule entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# description "store the RPID username in $username"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name Remote-Party-ID
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-value extract user store-as username
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# store-rule entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# description "store the privacy parameter in $repid-privacy"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name Remote-Party-ID
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-value extract parameter privacy store-as rpid_privacy
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# store-rule entry 3
Router(config-sbc-sbe-sip-hdr-ele-act)# description "store the From sip uri in $username"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and variable username is-defined eq false
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name From header-uri is-sip-uri eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name From header-uri sip-uri-user store-as username
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr)# store-rule entry 4
Router(config-sbc-sbe-sip-hdr-ele-act)# description "store the From tel uri in $username"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition is-request eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and variable username is-defined eq false
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name From header-uri is-tel-uri eq true
SIP Message Editing Using Editors

Router(config-sbc-sbe-sip-hdr-ele-act)# condition and header-name From header-uri
tel-uri-user store-as username
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# store-rule entry 5
Router(config-sbc-sbe-sip-hdr-ele-act)# description "convert RPID param into Privacy header value"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable rpid_privacy is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and variable rpid_privacy eq "off"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and "none" store-as privacy
Router(config-sbc-sbe-sip-hdr-ele-act)# exit
Router(config-sbc-sbe-sip-hdr-ele-act)# store-rule entry 6
Router(config-sbc-sbe-sip-hdr-ele-act)# description "convert RPID param into Privacy header value"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable rpid_privacy is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and variable rpid_privacy eq "id"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition and "user" store-as privacy
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header P-Preferred-Identity entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header P-Preferred-Identity entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# action add-first-header value "<sip:${username}@mydomain.com;user=phone>"
Router(config-sbc-sbe-sip-hdr-ele-act)# description "create a P-Preferred-Identity header"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable username is-defined eq true
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header P-Asserted-Identity entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header Remote-Party-ID entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header Privacy entry 1
Router(config-sbc-sbe-sip-hdr-ele-act)# action strip
Router(config-sbc-sbe-sip-hdr-ele-act)# exit

Router(config-sbc-sbe-sip-hdr-ele-act)# header Privacy entry 2
Router(config-sbc-sbe-sip-hdr-ele-act)# action add-first-header value "${privacy}"
Router(config-sbc-sbe-sip-hdr-ele-act)# description "create a privacy header if we have privacy info"
Router(config-sbc-sbe-sip-hdr-ele-act)# condition variable privacy is-defined eq true

Associate with an inbound adjacency:

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc test
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-sip)# header-editor inbound headprof1

Body Editor Example

The following example shows how to configure the beditor1 body editor on the SBC2 SBC:

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip body-editor beditor1
The following example shows how the `show sbc sbe sip body-editor` command is used to display details of the beditor1 body editor:

```
Router# show sbc SBC2 sbe sip body-editor beditor1

body-editor "beditor1"
  Description:
  Bodies:
    dtmf-relay/mixed
    action reject
    hunt-on-reject false
  Not in use with any adjacencies
  Not in use with any method-editor
```

**Example—Applying Body Editor**

The `body-editor inbound beditor1` command and the `body-editor outbound beditor1` command applies the beditor1 body editor on the inbound and outbound direction.

```
Router# configure terminal
Enter configuration commands, one per line.  End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip trans-uac
Router(config-sbc-sbe-adj-sip)# no attach
Router(config-sbc-sbe-adj-sip)# body-editor inbound beditor1
Router(config-sbc-sbe-adj-sip)# body-editor outbound beditor1
Router(config-sbc-sbe-adj-sip)# attach
```

The following examples shows how the `show sbc sbe sip body-editor` command is used to display details of the beditor1 body editor after it has been applied to an adjacency:

```
Router# show sbc SBC2 sbe sip body-editor
body-editors for SBC service "SBC2"
  Name                          In use
  =============================
  be1                            No
  beditor1                       Yes
  default                        No
```

```
Router# show sbc SBC2 sbe sip body-editor beditor1
body-editor "beditor1"
  Description:
  Bodies:
    dtmf-relay/mixed
    action reject
    hunt-on-reject false
  In use by adjacency:trans-uac (in, out)
  Not in use with any method-editor
```
Option Editor Example

The following example shows how to configure the oeditor1 option editor on the SBC2 SBC:

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip option-editor oeditor1
Router(config-sbc-sbe)# option opt

Example—Applying Option Editor

The option-editor inbound oeditor1 command and the option-editor outbound oeditor1 command applies the oeditor1 option editor on the inbound and outbound direction.

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip trans-uac
Router(config-sbc-sbe-adj-sip)# no attach
Router(config-sbc-sbe-adj-sip)# option-editor ua inbound oeditor1
Router(config-sbc-sbe-adj-sip)# option-editor ua outbound oeditor1
Router(config-sbc-sbe-adj-sip)# attach

The following shows how the show sbc sbe sip option-editor command is used to display details of the oeditor1 option editor:

Router# show sbc SBC2 sbe sip option-editor oeditor1
option-editor "oeditor1"
  Description:
  Type:    Whitelist
  Options:
  opt
  In use by adjacency:ASR-15 (in-ua)

Router# show sbc SBC2 sbe sip option-editor
option editors for SBC service "SBC2"
Name                     In use
==========================
opt                       No
 oeditor1                  Yes

Parameter Editor Example

The following example shows how to configure the peditor1 parameter editor on the SBC2 SBC:

Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip parameter-editor peditor
Router(config-sbc-sbe-mep-prm)# parameter param
Router(config-sbc-sbe-mep-prm-ele)# action strip
Example—Applying Parameter Editor

The following example shows how to apply the peditor parameter editor to the he1 header editor on the SBC2 SBC:

```plaintext
Router# configure terminal
Enter configuration commands, one per line. End with CNTL/Z.
Router(config)# sbc SBC2
Router(config-sbc)# sbe
Router(config-sbc-sbe)# sip header-editor he1
Router(config-sbc-sbe-mep-hdr)# header Subject
Router(config-sbc-sbe-mep-hdr-ele)# parameter-editor peditor
```

The following shows how the `show sbc sbe sip header-editor` command is used to display details of the he1 header editor:

```plaintext
Router# show sbc SBC2 sbe sip header-editor he1
header-editor "he1"
  Description:
    Type: Whitelist
    store-rules:
      No store-rule entries found.
    request-line:
      No request-line entries found.
    headers:
      subject
        entry 1
          description:
            action as-profile
            parameter-profile peditor
```

The following example shows how the `show sbc sbe sip parameter-editor` command is used to display details of the peditor parameter editor:

```plaintext
Router# show sbc SBC2 sbe sip parameter-editor peditor
parameter-editor "peditor"
  Description:
    Parameters:
      param
        action strip
    In use by header-editor:he1, header:subject, entry:1
```

SDP Editing Using Script-Based Editors

This section describes script-based editors for modifying the SDP content in SIP messages. The related SIP Message Editing Using Editors section on page 23-64 describes body, header, method, option, and parameter editors that you directly configure on the SBC. You can apply any combination of script-based editors and directly configured editors to edit SIP messages.

From Release 3.4S, you can use scripts written using the Lua programming language to modify the SDP content in SIP messages. Typically, a Lua script consists of a group of one or more related functions. In the context of the SIP Message Editing feature, you write these functions with the objective of editing SIP messages. For detailed information about the Lua programming language, visit the Lua website at http://www.lua.org/.
You can configure a set of Lua scripts on the SBC. A set of scripts describes a set of editing actions to be applied to SIP messages. While configuring a script set, you specify the order in which scripts are loaded in the script set.

You can register the message-editing functions in the scripts as editors. These editors are called by the SBC at run time and applied to SIP messages. You can use these editors in conjunction with the body, header, method, option, and parameter editors configured on the SBC.

After you configure a script set, you can perform isolation testing and live testing on the script set to ensure that it works as expected.

At any point of time, only one script set can be active on the SBC. However, multiple script sets can be defined and kept ready for future use. You can switch a script set from the active state to the inactive state according to your requirements and vice versa.

The following sections provide information about creating Lua scripts and configuring script-based editing:

- Creating Lua Scripts for Script-Based Editing, page 23-85
- Configuring Script-Based Editors on the SBC, page 23-91
- Creating and Configuring Script-Based Editors: Examples, page 23-99

Creating Lua Scripts for Script-Based Editing

The following sections provide information that you can use while creating Lua scripts for script-based editing:

- Built-in Lua Classes, page 23-85
- Built-in Application Variables, page 23-89
- Built-in Logger Functions, page 23-90
- Built-in Register Function, page 23-90
- User-Defined Application Variables, page 23-91

Built-in Lua Classes

Lua scripts use an XPath-compatible method of referring to each node within the SDP body of a SIP message. The following example shows a sample SDP body in XML format. In the Lua code that you write, each XML tag can be uniquely identified by its path. A syntax-based function (such as the MeBlock:select_by_syntax function that is explained in the ?$paranum>MeBlock Class? section on page 23-86) can accept and process data based on the path that is passed to the function. A path is a forward-slash-separated string. For example, the sdp/media[1]/line[3] path identifies the third line in the first media tag. Therefore, the sdp/media[1]/line[3] path refers to b=AS:64.

```
<sdp>
  <line>v=0</line>
  <line>c=user1 12345 12346 IN IP4 192.0.2.27</line>
  <line>o=-</line>
  <line>c=IN IP4 0.0.0.0</line>
  <line>t=0 0</line>
  <line>r=604800 3600 0 90000</line>
  <line>r=7d 1h 0 25h</line>
  <line>a=foo</line>
  <media>
    <line>m=audio 32768 RTP/AVP 0 101</line>
    <line>c=IN IP4 0.0.0.0</line>
  </media>
</sdp>
```
<line>b=AS:64</line>
<line>a=rtpmap:0 PCMU/8000</line>
<line>a=rtpmap:101 telephone-event/8000</line>
<line>a=ptime:20</line>
</media>
<media>
<line>m=video 32770 RTP/AVP 112</line>
<line>a=rtpmap:112 mpeg4-generic/48000</line>
</media>
</sdp>

You can use the following built-in Lua classes when writing scripts to modify the SDP body of SIP messages.

**MeMsg Class**

An object of the MeMsg class contains the top-level structure of the message, which in turn, contains the entire SIP message. Table 23-4 describes the functions of this class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:get_sdp() or .sdp</td>
<td>Returns the MeBlock object that holds the SDP body.</td>
</tr>
<tr>
<td>:get_current_edit_point</td>
<td>Returns the current edit point, which is either before-receive or after-send.</td>
</tr>
<tr>
<td>:reject(error_code)</td>
<td>Fails a SIP request, or discards the response.</td>
</tr>
<tr>
<td>:get_app_variables() or .app_variable</td>
<td>Returns the table of application variables.</td>
</tr>
</tbody>
</table>

**MeBlock Class**

An object of the MeBlock class represents a node in the SDP tree. A block is a contiguous subset of the SDP that is used for accessing strings. Table 23-5 describes the functions of the MeBlock class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.new(syntax)</td>
<td>Constructs a block using the given syntax.</td>
</tr>
<tr>
<td>:get_type() or .type</td>
<td>Returns the syntax type (line, media, or sdp) of the MeBlock object.</td>
</tr>
<tr>
<td>:get_parent() or .parent</td>
<td>Returns the parent of this MeBlock object, which is either another MeBlock object or NIL for the root.</td>
</tr>
<tr>
<td>:get_children() or .children</td>
<td>Returns a MeSelection object that contains the child elements of the block.</td>
</tr>
<tr>
<td>:select_by_prefix(text_prefix)</td>
<td>Returns a MeSelection object containing all the lines of the MeBlock object that have the specified prefix.</td>
</tr>
</tbody>
</table>
An object of the MeSelection class is a list of MeBlock objects. Objects of the MeSelection class are used to process a set of lines. They can also be used to process child blocks in a parent block. A MeSelection object sequences MeBlock objects in the order in which they appear in the message. Table 23-6 describes the functions of the MeSelection class.

**Table 23-6  Functions of the MeSelection Class**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:empty()</td>
<td>Returns TRUE if this selection is empty.</td>
</tr>
<tr>
<td>:iter()</td>
<td>Returns a generic For iterator that performs the specified action on all the objects in the MeSelection object. Each object is either of the MeBlock class or one of its subclasses.</td>
</tr>
<tr>
<td>[] operator</td>
<td>Use this one-based array operator to get the nth block in the MeSelection object. Negative array indexes are also supported.</td>
</tr>
</tbody>
</table>

**MeTextBlock Class**

An object of the MeTextBlock class is used to assign, create, or manipulate text. Table 23-7 describes the functions of this class.

**Table 23-7  Functions of the MeTextBlock Class**

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.new(type,text)</td>
<td>Constructs a new block of a specific type (line, media, and so on) using the specified text.</td>
</tr>
<tr>
<td>:get_text() or .text</td>
<td>Returns the text of the MeTextBlock object.</td>
</tr>
</tbody>
</table>
### MeSdp Class

An object of the MeSdp class is used to retrieve specific parts of the SDP body. Table 23-8 describes the functions of this class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>:get_session_lines() or .session_lines</td>
<td>Returns a MeSelection object containing the session lines of the SDP body.</td>
</tr>
<tr>
<td>:get_media_blocks() or .media_blocks</td>
<td>Returns a MeSelection object containing the media blocks.</td>
</tr>
</tbody>
</table>

### MeSdpMedia Class

An object of the MeSdpMedia class is used to create or retrieve SDP media blocks. Table 23-9 describes the functions of this class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.new(text)</td>
<td>Constructs a block of the media syntax (or block type) using the specified text.</td>
</tr>
<tr>
<td>:get_media_lines() or .media_lines</td>
<td>Returns a MeSelection object containing the media lines of the MeSdpMedia object.</td>
</tr>
</tbody>
</table>
MeSdpLine Class

An object of the MeSDPLine class is used to create a line in the SDP message. Table 23-10 describes the functions of this class.

<table>
<thead>
<tr>
<th>Function</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>.new(text)</td>
<td>Constructs a block of the line syntax (or block type) using the specified text.</td>
</tr>
</tbody>
</table>

Built-in Application Variables

This section describes the built-in application variables that you can use while writing Lua scripts. Built-in application variables, such as configuration data for an adjacency and transport values, are initialized by the SBC and are available to the Lua scripts. They are read-only, start with the characters msg. or adj., and are reserved because you cannot create variables with these prefixes. Table 23-11 describes the built-in application variables that can be accessed within a script.

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>adj.account</td>
<td>Adjacency account.</td>
</tr>
<tr>
<td>adj.dest_addr</td>
<td>Adjacency signaling peer.</td>
</tr>
<tr>
<td>adj.dest_port</td>
<td>Adjacency signaling peer port.</td>
</tr>
<tr>
<td>adj.group</td>
<td>Adjacency group.</td>
</tr>
<tr>
<td>adj.home_net_id</td>
<td>Adjacency home network identity.</td>
</tr>
<tr>
<td>adj.ip_realm</td>
<td>Adjacency realm.</td>
</tr>
<tr>
<td>adj.lcl_addr</td>
<td>Adjacency signaling address.</td>
</tr>
<tr>
<td>adj.lcl_port</td>
<td>Adjacency signaling port.</td>
</tr>
<tr>
<td>adj.lcl_id</td>
<td>Adjacency local ID.</td>
</tr>
<tr>
<td>adj.listen_trans</td>
<td>Adjacency listen transport.</td>
</tr>
<tr>
<td>adj.mandatory_trans</td>
<td>Adjacency mandatory transport.</td>
</tr>
<tr>
<td>adj.med_loc</td>
<td>Adjacency media location.</td>
</tr>
<tr>
<td>adj.name</td>
<td>Adjacency name.</td>
</tr>
<tr>
<td>adj.preferred_trans</td>
<td>Adjacency preferred transport.</td>
</tr>
<tr>
<td>adj.trust_level</td>
<td>Adjacency trust level.</td>
</tr>
<tr>
<td>adj.target_reg_addr</td>
<td>Adjacency registration target address.</td>
</tr>
<tr>
<td>adj.targrt_reg_port</td>
<td>Adjacency registration target port.</td>
</tr>
<tr>
<td>adj.visited_net_id</td>
<td>Adjacency visited network identity.</td>
</tr>
<tr>
<td>adj.vpn_id</td>
<td>Adjacency VPN ID.</td>
</tr>
</tbody>
</table>
Table 23-11  Built-in Application Variables

<table>
<thead>
<tr>
<th>Variable</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>msg.status_code</td>
<td>Response status code. Returns the string representation of the status code for a SIP response message. Returns an empty string for a SIP request message.</td>
</tr>
<tr>
<td>msg.header(&quot;name&quot;).value</td>
<td>Value of the first header with the name name in the message. Only nonvital headers can be used with this function.</td>
</tr>
<tr>
<td>msg.header(&quot;name&quot;).uri.tel_uri.number</td>
<td>Directory number of the TEL URI in the first header with the name name. If used on a SIP URI, an empty string is returned. Only To and From headers can be used with this function.</td>
</tr>
<tr>
<td>msg.header(&quot;name&quot;).uri.sip_uri.user</td>
<td>User name of the SIP URI or SIPS URI in the first header with the name name. If used on a TEL URI, an empty string is returned. Only To and From headers can be used with this function.</td>
</tr>
<tr>
<td>msg.lcl_ip_addr</td>
<td>Address at which the message was received.</td>
</tr>
<tr>
<td>msg.lcl_port</td>
<td>Port at which the message was received.</td>
</tr>
<tr>
<td>msg.rmt_ip_addr</td>
<td>Previous hop IP address.</td>
</tr>
<tr>
<td>msg.rmt_port</td>
<td>Previous hop port.</td>
</tr>
</tbody>
</table>

Built-in Logger Functions

The following logger functions can be called to create logs:
- MeLogger.debug(text) Log at debug level (30)
- MeLogger.detail(text) Log at detail level (50)
- MeLogger.info(text) Log at info level (60)
- MeLogger.config(text) Log at config level (63)
- MeLogger.warn(text) Log at warn level (70)
- MeLogger.error(text) Log at error level (80)

Built-in Register Function

Use the following function to register functions as editors with the SBC:
MeEditor.register(edit_point,editor_name,edit_func)

By including this line in the script, you can register a function as an editor with the SBC, assign the function a name as an editor, and specify the point at which the function must be applied as an editor on SIP messages.

The following are the arguments of the MeEditor.register function:
- edit_point—Accepts one of the following values:
  - AFTER_SEND—Specifies that the outgoing message must be edited after it is processed by the adjacency and just before it is forwarded from the adjacency.
– **BEFORE_RECEIVE**—Specifies that the incoming message must be edited just after it is received on the adjacency and before the adjacency begins processing it.

- **editor_name**— Specifies the name that you want to assign to the editor.

**Note** Names that you assign to editors in a script set must be unique. However, editors in different script sets can have the same name.

- **edit_func** is the name of the function in the script that you want to designate as an editor.

The following example shows how to register the `hello_world` Lua function as an editor:

```python
MeEditor.register(MeEditor.BEFORE_RECEIVE,
    "hello_world_editor",
    hello_world)
```

### User-Defined Application Variables

User-defined application variables are used to pass user information among Lua edit functions and between script-based editors and editors that are directly configured on the SBC that is, body, header, method, option, and parameter editors.

### Configuring Script-Based Editors on the SBC

This task shows how to configure a script-based editor on the SBC.

**Note** Before you start performing this task, create the scripts offline and place the script files at a location from where they can be accessed from the SBC. Copy the script files to the SBC by using trivial file transfer protocol (TFTP), file transfer protocol (FTP), remote copy protocol (rcp), secure copy protocol (SCP), or any other supported application.

### SUMMARY STEPS

1. `configure terminal`
2. `sbc sbc-name`
3. `sbe`
4. `script-set set-number lua`
5. `script script-name`
6. `load-order order-index-number`
7. `type {full | wrapped} edit-point {before-receive | after-send | both}
8. `filename {bootflash: | flash: | fpd: | nvram: | obfl: | any-other-device}
9. `exit`
10. `complete`
11. `end`
12. `test sbc message sip filename device-type:file-name script-set script-set-number {after-send | before-receive} editors {editor1-name [editor2-name] [editor3-name]...[editor8-name]}`
13. configure terminal
14. sbc sbc-name
15. sbe
16. adjacency sip adjacency-name
17. test script-set set-number
18. exit
19. active-script-set script-set-number
20. adjacency sip adjacency-name
21. editor-list {after-send | before-receive}
22. editor order-number editor-name [condition [body contains sdp]]
23. end
24. show sbc sbc-name sbe script-set set-number [script script-name [line-numbers] | program [line-numbers] | statistics]
25. clear sbc sbc-name sbe script-set-stats set-number [editors-stats editor-name]
## DETAILED STEPS

<table>
<thead>
<tr>
<th>Step 1</th>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>configure terminal</td>
<td>Enters the global configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router# configure terminal</td>
<td></td>
</tr>
<tr>
<td>Step 2</td>
<td>sbc sbc-name</td>
<td>Enters the SBC service mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config)# sbc mysbc</td>
<td></td>
</tr>
<tr>
<td>Step 3</td>
<td>sbe</td>
<td>Enters the SBE configuration mode.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc)# sbe</td>
<td></td>
</tr>
<tr>
<td>Step 4</td>
<td>script-set set-number lua</td>
<td>Configures a script set composed of scripts written using the Lua programming language.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe)# script-set 20 lua</td>
<td></td>
</tr>
<tr>
<td>Step 5</td>
<td>script script-name</td>
<td>Configures a script in the script set. Note that multiple scripts can be configured in a script set.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe-script-set)# script SBCScript</td>
<td></td>
</tr>
<tr>
<td>Step 6</td>
<td>load-order order-index-number</td>
<td>Specifies the load order of the script. Scripts are loaded in ascending order of the order index number. For example, a script with the order index number 4 is loaded before a script with the order index number 6.</td>
</tr>
<tr>
<td></td>
<td><strong>Example:</strong></td>
<td></td>
</tr>
<tr>
<td></td>
<td>Router(config-sbc-sbe-scrpset-script)# load-order 2</td>
<td></td>
</tr>
</tbody>
</table>

### Additional Information

- **sbc-name**—Name of the SBC.

- **set-number**—Script set number. The range is from 1 to 2147483647.

- **order-index-number**—Load order index number. The range is from 1 to 4294967295. The default order index number is 100. For scripts that are subsequently added and for which the load-order command is not run, the default order index number is set in multiples of 100 (that is, 200, 300, 400, and so on).
## Command or Action

| Step 7 | type {full | wrapped edit-point (after-send | before-receive | both)} |
|--------|--------------------------------------------------|
| Example: | Router(config-sbc-sbe-scrpset-script)# type full |

Specifies the script type:
- **full**—Specifies a full script without autogeneration.
- **wrapped edit-point**—Specifies a script that must be autogenerated from the file and the edit point to be used in autoregistration. One of the following edit points can be specified:
  - **after-send**—Specifies that the outgoing message must be edited after the message is processed by the adjacency and just before it is forwarded from the adjacency.
  - **before-receive**—Specifies that the incoming message must be edited just after it is received on the adjacency and before the adjacency begins processing it.
  - **both**—Enables editing of the message both before and after it is processed by the SBC.

<table>
<thead>
<tr>
<th>Step 8</th>
<th>filename {device-type:file-path-and-name}</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-scrpset-script)# filename bootflash:lua1.lua</td>
</tr>
</tbody>
</table>

Specifies the path and name of the script file.
- **device-type**—One of the following or any other storage device installed on the router:
  - `bootflash:`
  - `flash:`
  - `fpd:`
  - `nvram:`
  - `obfl:`

The list of file system devices is dynamically generated and displayed. Other devices, such as a hard disk, that are available on the platform can also be used in this command.
- **file-path-and-name**—Full path and name of the script file on the specified storage device.

<table>
<thead>
<tr>
<th>Step 9</th>
<th>exit</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-scrpset-script)# exit</td>
</tr>
</tbody>
</table>

Exits the script configuration mode and enters the script-set configuration mode.

<table>
<thead>
<tr>
<th>Step 10</th>
<th>complete</th>
</tr>
</thead>
<tbody>
<tr>
<td>Example:</td>
<td>Router(config-sbc-sbe-script-set)# complete</td>
</tr>
</tbody>
</table>

Validates and loads the scripts.
If syntax errors are encountered during the validation process, error messages are displayed. If a script is syntactically correct, it is loaded into memory and the editors are registered with the Lua run-time environment. If required, you can switch to the privileged EXEC mode and then run the `show sbc sbe editors` command to verify that the editors are correctly registered.
**Chapter 23  SIP Message Manipulation**

**SDP Editing Using Script-Based Editors**

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td>Step 11  end</td>
<td>Exits the script-set configuration mode, and returns to the privileged EXEC mode.</td>
</tr>
</tbody>
</table>

**Example:**
Router(config-sbc-sbe-script-set)# end

Step 12  test sbc message sip filename device-type:file-name script-set script-set-number (after-send | before-receive) editors {editor1-name [editor2-name] [editor3-name] . . . [editor8-name]}

**Example:**
Router# test sbc message sip filename bootflash:inv script-set 123 after-send editors sdp_add_after my-header-editor

Performs isolation testing of script-based editors.

**Note**  
Although it is optional to perform isolation testing, we recommend that you perform the procedure. See the ?$paranum>Isolation Testing of Script-Based Editors: Example? section on page 23-100 for detailed information about the procedure.

- **device-type**—Type of storage device on which you have stored the file containing the SIP message on which you want to test the editors. In the command-line interface (CLI), when you enter a question mark after the test sbc message sip filename script-set editors command, a list of all the storage devices installed on the router is displayed. The device can be one of the following or any other storage device installed on the router:
  - bootflash:
  - flash:
  - fpd:
  - nvram:
  - obfl:
  
The list of file system devices is dynamically generated and displayed. Other devices, such as a hard disk, that are available on the platform can also be used in this command.

- **file-name**—Name of the file containing the SIP message on which you want to test the editors.

- **script-set-number**—Number of the script set containing the editors that you want to test.

- **after-send**—Specifies that the outgoing message must be edited after the message is processed by the adjacency and just before it is forwarded from the adjacency.

- **before-receive**—Specifies that the incoming message must be edited just after it is received on the adjacency and before the adjacency begins processing it.

- **editor1-name . . . editor8-name**—Names of the editors. You can specify up to eight editors. You must specify at least one editor.
### Command or Action | Purpose
--- | ---
**Step 13** configure terminal | Enters the global configuration mode.
**Example:**
Router# configure terminal

**Step 14** sbc sbc-name | Enters the SBC service mode.
- **sbc-name**—Name of the SBC.
**Example:**
Router(config)# sbc mysbc

**Step 15** sbe | Enters the SBE configuration mode of the SBC.
**Example:**
Router(config-sbc)# sbe

**Step 16** adjacency sip adjacency-name | Enters the SBE SIP adjacency configuration mode.
- **adjacency-name**—Name of the adjacency.
**Example:**
Router(config-sbc-sbe)# adjacency sip adj1

**Step 17** test script-set script-set-number | Performs live testing of script-based editors.
**Note** Although it is optional to perform live testing, we recommend that you perform the procedure. See the **Live Testing of Script-Based Editors: Example** section on page 23-102 for detailed information.
- **script-set-number**—Script set number.
**Example:**
Router(config-sbc-sbe-adj-sip)# test script-set 123

**Step 18** exit | Exits the SIP adjacency configuration mode.
**Example:**
Router(config-sbc-sbe-adj-sip)# exit

**Step 19** active-script-set script-set-number | Activates the script set.
- **script-set-number**—Script set number.
**Note** When you run the **active-script-set** command for a particular script set, the script set that was previously active automatically goes to the inactive state. The editors of an inactive script set are no longer applied to SIP messages. You can switch an inactive script set to the active state by running the **active-script-set** command on it.
**Example:**
Router(config-sbc-sbe)# active-script-set 20

**Step 20** adjacency sip adjacency-name | Enters the SBE SIP adjacency configuration mode.
- **adjacency-name**—Name of the adjacency.
**Example:**
Router(config-sbc-sbe)# adjacency sip adj1
## Command or Action

**Step 21**  
```
editor-list (after-send | before-receive)
```

**Example:**
```
Router(config-sbc-sbe-adj-sip)# editor-list
after-send
```

**Purpose:** Configures a list of editors.
- **after-send**—Specifies that the outgoing message must be edited after the message is processed by the adjacency and just before it is forwarded from the adjacency.
- **before-receive**—Specifies that the incoming message must be edited just after it is received on the adjacency and before the adjacency begins processing it.

**Step 22**  
```
editor order-number editor-name [condition [body contains sdp]]
```

**Example:**
```
Router(config-sbc-sbe-adj-sip-ed)# editor 2
sdp_add_after condition body contains sdp
```

**Purpose:** Configures an editor in the editor list. For each editor that you want to apply in a sequence, run this command to specify the order of the editor in the editor list.

**Note** You can add any combination of script-based editors and body, header, method, option, and parameter editors in the editor list.
- **order-number**—Order in which the editor must be applied. The range is from 1 to 2147483647.
- **editor-name**—Name of the editor that you want to apply to messages that are processed by the adjacency.
- **condition**—Specifies that there are one or more conditions for the editor to be applied.
- **body contains sdp**—Specifies that the message body must be SDP-based content. The editor is applied only if this condition is met. Include `body contains sdp` in the command for script-based editors.

**Step 23**  
```
end
```

**Example:**
```
Router(config-sbc-sbe-adj-sip)# end
```

**Purpose:** Exits the SIP editor configuration mode, and enters the privileged EXEC mode.
Chapter 23  SIP Message Manipulation

SDP Editing Using Script-Based Editors

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 24</strong></td>
<td>Displays a summary of the details pertaining to all the configured script sets or shows the details of the specified script set.</td>
</tr>
<tr>
<td>`show sbc sbc-name sbe script-set set-number [script script-name [line-numbers]</td>
<td>program [line-numbers]</td>
</tr>
<tr>
<td>Example:</td>
<td><code>Router# show sbc mysbc sbe script-set 20 script SBCscript line-numbers</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Command or Action</th>
<th>Purpose</th>
</tr>
</thead>
<tbody>
<tr>
<td><strong>Step 25</strong></td>
<td>Clears previously stored statistics related to a script set.</td>
</tr>
<tr>
<td><code>clear sbc sbc-name sbe script-set-stats</code></td>
<td></td>
</tr>
<tr>
<td><code>script-set-number [editors-stats editor-name]</code></td>
<td></td>
</tr>
<tr>
<td>Example:</td>
<td><code>Router# clear sbc mysbc sbe script-set-stats 1</code></td>
</tr>
</tbody>
</table>

The following example shows how the `show sbc sbe script-set` command is used to display the summary of a script set:

```bash
Router# show sbc mysbc sbe script-set 1
name         language complete active status
------------- ----------------- ------- ------- -------
script-set 1  lua       yes    no       ok

script         order   filename
-------------------------
edit_invite_1   1       bootflash:lua_1.lua
edit_invite_2   2       bootflash:lua_2.lua
edit_invite_3   3       bootflash:lua_3.lua
```
Creating and Configuring Script-Based Editors: Examples

The following sections describe how to create and configure sample script-based editors:

- Creating Lua Scripts: Example, page 23-99
- Configuring Script-Based Editors: Example, page 23-100
- Isolation Testing of Script-Based Editors: Example, page 23-100
- Live Testing of Script-Based Editors: Example, page 23-102

Creating Lua Scripts: Example

The following sections provide listings of sample Lua scripts:

- Adding Text in the SDP Body: Example, page 23-99
- Deleting Lines from the SDP Body: Example, page 23-100
- Replacing Text in the SDP Body: Example, page 23-100

Adding Text in the SDP Body: Example

The following example shows a Lua script that is used to add `sdp_add_after added this line` at the end of the SDP body:

```lua
function append_text(msg)
    msg.sdp:insert_child_last(MeSdpLine.new("sdp_add_after added this line"))
end
```

The following example shows the line that registers the `append_text` Lua function from the preceding example as an editor. In this example, the editor is named `sdp_add_after`.

```lua
MeEditor.register(MeEditor.BEFORE_RECEIVE, "sdp_add_after", append_text)
```

Note

An editor is registered with the SBC when the script set containing the script with the editor code is configured on the SBC.

The complete code listing for this script is as follows:

```lua
function append_text(msg)
    msg.sdp:insert_child_last(MeSdpLine.new("sdp_add_after added this line"))
end
MeEditor.register(MeEditor.BEFORE_RECEIVE, "sdp_add_after", append_text)
```

You can save these lines in a `.lua` file, copy the file to the SBC, and then perform the procedure described in the Configuring Script-Based Editors on the SBC? section on page 23-91 to configure and test the editor.
Deleting Lines from the SDP Body: Example

The following script deletes all the lines that start with \texttt{a=deleteme} from the SDP bodies of SIP messages:

```javascript
function delete_lines(msg)
    for line in msg.sdp:select_by_prefix("a=deleteme"):iter() do
        line:delete()
    end
end
MeEditor.register(MeEditor.BEFORE_RECEIVE, "Delete_a_Lines", delete_lines)
```

Replacing Text in the SDP Body: Example

The following script replaces \texttt{rtpmap} in the SDP body with \texttt{srtp_remap}:

```javascript
function replace_text(msg)
    msg.sdp:replace("rtpmap", "srtp_remap")
end
MeEditor.register(MeEditor.AFTER_SEND, "Switch_Protocol", replace_text)
```

Configuring Script-Based Editors: Example

The following example shows how to configure the script set created in the preceding example:

```bash
Router# configure terminal
Router(config)# sbc mysbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# script-set 20 lua
Router(config-sbc-sbe-script-set)# script SBCScript
Router(config-sbc-sbe-scpset-script)# load-order 2
Router(config-sbc-sbe-scpset-script)# type full
Router(config-sbc-sbe-scpset-script)# filename bootflash:lua1.lua
Router(config-sbc-sbe-scpset-script)# exit
Router(config-sbc-sbe-script-set)# complete
Router(config-sbc-sbe-script-set)# end
Router# test sbc message sip filename bootflash:inv script-set 123 after-send editors
sdp_add_after my-header-editor
Router# configure terminal
Router(config)# sbc mysbc
Router(config-sbc)# sbe
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# test script-set 123
Router(config-sbc-sbe-adj-sip)# exit
Router(config-sbc-sbe)# active-script-set 20
Router(config-sbc-sbe)# adjacency sip adj1
Router(config-sbc-sbe-adj-sip)# editor-list after-send
Router(config-sbc-sbe-adj-sip-adj-sip-ed)# editor 2 sdp_add_after condition body contains sdp
Router(config-sbc-sbe-adj-sip-adj-sip)# end
Router# show sbc mysbc sbe script-set 20 script SBCscript line-numbers
```

Isolation Testing of Script-Based Editors: Example

During isolation testing of script-based editors, the SDP editing configuration is tested in isolation. No other form of SBC processing takes place. Isolation testing does not show interactions between the editing configuration and other configurations, such as, number validation configuration.

The \texttt{test sbc message} command is used to perform isolation testing on SIP messages. This command loads a file containing a valid protocol message and applies a list of user-specified editors to the message. It does not display details of interactions between editing and routing decisions. Up to eight editors can
be specified in the command. The order in which the editors are specified is the order in which they are applied. Note that profile editors that are not part of any specific script set can also be specified in the command.

In the following example, sdp_add_after is defined in script-set 123 and my_header_editor has been configured using the `sip header-editor` command. The sdp_add_after editor is the one used in the preceding sections describing examples. The lines highlighted in bold show the actions performed by the editors.

```
Router# test sbc message sip filename bootflash:inv script-set 123 after-send editors sdp_add_after my-header-editor

INVITE sip:john@example.com:55060 SIP/2.0
Via: SIP/2.0/UDP 192.0.2.195;branch=z9hG4bKff9b46fb055c0521cc240244a96cd290
Via: SIP/2.0/UDP 192.0.2.195:55061;branch=z9hG4bK291d90e31a47b225bd0ddff4353e9c0
From: <sip:192.0.2.195:55061;user=phone>;tag=GR52RW346-34
To: "john@example.com" <sip:john@example.com:55060>
Call-ID: 12013223@192.0.2.195
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: 229

name: cisco
v=0
o=Clarent 120386 120387 IN IP4 192.0.2.196
s=Clarent CSCM
c=IN IP4 192.0.2.196
t=0 0
m=audio 40376 RTP/AVP 8 18 4 0
a=rtpmap:8 PCMA/8000
a=rtpmap:18 G729/8000
a=rtpmap:4 G723/8000
a=rtpmap:0 PCMU/8000
a=SendRecv

sdp_add_after added this line
```
Live Testing of Script-Based Editors: Example

During live testing of script-based editors, an adjacency is configured as a test adjacency. Inbound editing and outbound editing of messages on that adjacency are then performed using the script set specified in the `test script-set` command instead of the script set that is currently active. The following is a sample command:

```
Router(config-sbc-sbe-adj-sip)# test script-set 123
```

**Note**

The active script set is specified by the `active-script-set` command. You must ensure that the `active-script-set` command has not been run on the script set on which you run the `test script-set` command.

The `test script-set` command cannot be used to verify profile editors because the profile editors are not associated with a script set. To include a profile editor in the test, first configure the profile editor on the test adjacency by using the `editor-list` command.