Reference Guide

AudioCodes Gateway & Session Border Controller Series

SIP Message Syntax

Message Manipulation | Message Conditions | Pre-Parsing Manipulation | Call Setup Rules

Version 7.4

audiocodes

Notice

Date Published: December-12-2024

WEEE EU Directive

Pursuant to the WEEE EU Directive, electronic and electrical waste must not be disposed of with unsorted waste. Please contact your local recycling authority for disposal of this product.

Security Vulnerabilities

All security vulnerabilities should be reported to vulnerability@audiocodes.com.

Customer Support

Customer technical support and services are provided by AudioCodes or by an authorized AudioCodes Service Partner. For more information on how to buy technical support for AudioCodes products and for contact information, please visit our website at https://www.audiocodes.com/services-support/maintenance-and-support.

Documentation Feedback

AudioCodes continually strives to produce high quality documentation. If you have any comments (suggestions or errors) regarding this document, please fill out the Documentation Feedback form on our website at https://online.audiocodes.com/documentation-feedback.

Stay in the Loop with AudioCodes

Related Documentation

MP-1288 High-Density Analog Media Gateway User's Manual
Mediant 500 Gateway & E-SBC User's Manual
Mediant 500L Gateway & E-SBC User's Manual
Mediant 800 Gateway & E-SBC User's Manual
Mediant 1000B Gateway & E-SBC User's Manual
Mediant 3100 Gateway & SBC User's Manual
Mediant 2600 E-SBC User's Manual
Mediant 4000 SBC User's Manual
Mediant 9000 SBC User's Manual
Mediant Software SBC User's Manual

Document Revision Record

LTRT	Description
29049	Initial document release for Version 7.4
29051	Typo in HTTP POST and GET Requests
29053	Max. characters and note for registered user variables
29055	Body part header manipulation added
29057	Typo in Message Type description
29058	Typo in regex detailed example; var.call.src|dst.TenantId added
29060	New SDP parameters (param.message.sdp.originusername / param.message.sdp.ip-for-url); param.message.sdp.address replaced by param.message.sdp.ip
29063	Call.key added; UUID-Generate added; ReferredByTags added; IP Group parameter syntax updated

LTRT	Description
29064	Updated to Ver. 7.40A.300 (Func.Encrypt; Func.Decrypt)
29065	IP Groups table parameter syntax param.ipg.dst|src updated
29066	Updated to Ver. 7.40A.500; max. specific unknown headers; Var.Call.Dst|Src.PlayBackgroundTone; call Variable for CDR value from SIP header (Gateway)
29067	Message normalization re URL updated; typo for "header.from.user" and "header.p-asserted-id.host"
29068	Updated to 7.4.500-2 (Call Variable for Maximum Call Duration)
29070	Updated to 7.4.600; logical operators AND and OR for Condition updated; max. sub-variables per rule increased to 15 (only Mediant Software 8 GB and higher); numerical comparisons for conditions using 'num'
29071	Supported SIP methods for X-AC-Action header
29072	Authorization header added to Detailed Header Syntax section; example of Allow header added to 'Header Examples' section.
29703	Updated to 7.4.500-5; 'cac' tag; update to rand.number/rand.string syntax; note re SBC manipulation ignored when routed to Gateway application

Table of Contents

1 Introduction

This document describes the SIP message syntax that is used for the following configuration tables:

Table 1-1: Configuration Tables and Relevant Fields

Table	Fields
Message Manipulations Table	■ 'Message Type' ■ 'Condition' ■ 'Action Subject' ■ 'Action Type' ■ 'Action Value'
Message Conditions Table	■ 'Condition'
Pre-Parsing Manipulation Table	■ 'Message Type' ■ 'Pattern' ■ 'Replace-With'
Call Setup Rules Table	■ 'Condition' ■ 'Action Subject' ■ 'Action Type' ■ 'Action Value'

For hybrid SBC-Gateway products: If you have configured call routing from the device's SBC application (IP-to-IP routing) to the device's Gateway application for IP-to-Tel routing, the device uses the initial SIP message as if it's a new call. Therefore, if any manipulations were done on the SIP message by the SBC application, the device ignores them.

2 Field Syntax

Message Type Field

The 'Message Type' field defines the type of SIP message that you want to apply the manipulation or condition rule.

Syntax:

```
<SIP-method/any>.<request/response/any>.<response-type>
```

where:

- <SIP-method/any> specifies the SIP method used with the option to specify requests of all method types.
- <request/response/any> specifies the SIP request or SIP response type with the option to specify any request or response type.
- <response-type> specifies the SIP response type. You can also use the 'x' wildcard to denote multiple response types:
  - To denote all SIP 18x responses (e.g., 180, 181, 182 and 183), use the following syntax: 18x
  - To denote all response types belonging to a specific response group (i.e., 1xx for provisional, 2xx for successful, 3xx for redirection, 4xx for client failure, 5xx for server failure, and 6xx for global failure responses), use two 'x' wildcards instead of the last two digits of the response: <first digit of response group>xx (e.g., 1xx)

For more information, see Message Type on page 197.

Message Type Examples

The following table provides examples of different message types.

Table 2-1: Message Type Examples

Message Types	Description
`invite.request`	INVITE requests
`invite.response.200`	INVITE 200 responses only
`register.response.2xx`	All 2xx responses for REGISTER
`invite.response.18x`	All 18x responses for INVITE
`subscribe.request`	All SUBSCRIBE requests

Message Types	Description
subscribe.response	All SUBSCRIBE responses
reinvite.request	re-INVITE requests
any.request	Requests of all method types, where any is a keyword.
any.response.200	All 200 responses for all method types, where any is a keyword.
invite	Requests and responses of INVITE method.
<empty>	All request and responses for all method types.
info.any	All INFO requests and responses.
private1.request	All requests with method 'private1'.

Condition Field

The 'Condition' field is used to test specific parts of the SIP header in the message with specified values. Conditions may be combined with other conditions using logical operators (and/or). If the condition is met, the device performs a configured action.

Syntax:

```
<subject> <operand> <value>
```

where:

- <subject> specifies the subject of the condition using the following format:
  header/body/parameter
- <operand> specifies the operand of the condition using the following format:
  condition-operand
- <value> specifies the value of the condition using the following format:
  string/header/body/parameter/random/variable/regex

For more information, see Condition on page 198.

Condition Field Operands

The following table describes the condition operands.

Table 2-2: Condition Operands

Condition Operand	Description
== !=	Tests for equivalent values. Tests for not equivalent values.
>= <=	Tests for greater than or equal to values. Tests for less than or equal to values.
> <	Tests for greater than values. Tests for less than values.
contains !contains	Tests a string containing specified text. Tests a string not containing specified text.
exists !exists	Tests whether a parameter exists. Tests whether a parameter does not exist.
suffix prefix	Tests whether a string has a particular suffix. Tests whether a string has a particular prefix.
len> len< len==	Tests whether the length of a string is greater than a specific value. Tests whether the length of a string is less than a specific value. Tests whether the length of a string is equal to a specific value.
num< num< num= num== num>= num<= num!=	Tests the numerical value (not lexicographic comparison) of the string. For more information, see Condition on page 198.
regex	Tests whether a string matches the given regular expression.
+	Concatenates string values.
insubnet !insubnet	Tests whether the host IP address (IPv4 or IPv6) in a SIP header (e.g., From or To) belongs to a specific subnet expressed in CIDR (Classless Inter-Domain Routing) notation. Tests whether the host IP address (IPv4 or IPv6) in a SIP header (e.g., From or To) does not belong to a specific subnet expressed in CIDR (Classless Inter-Domain Routing) notation.

Condition Field Examples

The following table provides examples of conditions.

Table 2-3: Condition Examples

Condition	Description
`header.expires.time < '88888'`	Returns true if expires time is less than '88888'.
`header.user-agent contains 'Android-VMAS' --OR-- header.user-agent contains 'MP252'`	Returns true if the user agent is 'Android-VMAS' or 'MP252'.
`param.message.sdp.ip == '10.132.10.101'`	Returns true if the "c=" line contains the given IP address.
`header.request-uri.methodtype=='415'`	Returns true if the message method type is '415'.
`header.diversion.0 regex (<.*)(;urlparam=[a-z]*)(.*>)`	Returns true if the REGEX engine matches urlparam=<specific value>.
`ldap.attr.msRTCSIP-Line contains 'tel:'+param.call.dst.user+':ext='`	LDAP Attribute contains three values, which are separated by the "+" operator.
`header.from.url.host insubnet '10.8.0.0/8'`	Returns true if the subnet of the IPv4 host in the From header is in the given subnet.
`header.from.url.host !insubnet 'ffff:a08:705:0:0::/32'`	Returns true if the IPv6 subnet of the host in the From header is not in the given subnet.
`header.to.url.user > '20'`	Returns true if user part in To header is lexicographically greater than 20.
`header.to.url.user num> '20'`	Returns true if user part in To header is numerically greater than 20.

Action Subject Field

The 'Action Subject' field defines the message component upon which you wish to manipulate.

Syntax:

- header
- param
- body
- variable
- message

For more information, see Action Subject on page 200.

Action Subject Field Examples

The following table provides various example actions.

Table 2-4: Action Examples

Action Subject	Action Type	Action Value	Description
`header.customername`	**Add**	`'Audiocodes'`	Adds the "customername" header to the message with a value of "Audiocodes".
`header.customername`	**Remove**		Deletes the header "customername" from the message.
`var.global.0`	**Modify**	`header.user-agent`	Stores the content of the User-agent header in a global variable. Note, the Modify action is executed on the variables (not the Add action).
`header.contact.param.company`	**Add**	`'audiocodes'`	Adds a parameter "company" to a

Action Subject	Action Type	Action Value	Description
			Contact header and assigns the value "Audiocodes" to it.

Action Type Field

The 'Action Type' field specifies the type of action you wish to perform on the message component:

Table 2-5: Action Type Field Options

Action Operand	Description
Add	Adds entities to a message.
Remove	Removes entities from a message.
Modify	Modifies parts of a header or SDP.
Add Prefix	Adds a string prefix to part of a header.
Add Suffix	Adds a string suffix to part of a header.
Remove Prefix	Removes a string prefix from part of a header.
Remove Suffix	Removes a string suffix to part of a header.

For more information, see Action Type on page 202.

Action Value Field

The 'Action Value' field defines the value to assign to the 'Action Type' and 'Action Subject'.

Syntax:

- string
- header
- body
- parameter
- random
- variable

- regex

For more information, see Action Value on page 203.

Action Value Field Examples

The following table provides various example actions.

Table 2-6: Action Examples

Action Subject	Action Type	Action Value	Description
`header.customername`	**Add**	`'ABCompany'`	Adds the "customername" header to the message with the value "ABCompany".
`header.customername`	**Remove**		Deletes the header "customername" from the message.
`var.global.0`	**Modify**	`header.user-agent`	Stores the content of the User-agent header in a global variable. Note, the Modify action is executed on the variables (not the Add action).
`header.contact.param.company`	**Add**	`'ABCompany'+'Sales'`	Adds the parameter "company" to the Contact header with the values "ABCompany" and "Sales".

3 Detailed Syntax

This section describes the detailed syntax usage of the fields in the Message Manipulations table. The following syntax is described:

- Strings – see Strings below
- Headers – see Headers on page 11
- Body – see Body on page 36
- Parameters – see Parameters on page 39

Strings

The string type is a series of characters and the most basic of all syntax types.

Syntax:

- String enclosed by a single apostrophe:

```
'string'
```

- To concatenate strings, use the plus "+" operator:

```
'string' + 'string'
```

- To indicate carriage returns (new lines), use the double-backslash (\\):

```
'string\\string'
```

Strings can be used as the value in the following fields:

Table 3-1: Configuration Tables and Relevant Fields for Strings

Table	Fields
Message Manipulations Table	■ 'Condition' ■ 'Action Value'
Message Conditions Table	■ 'Condition'
Pre-Parsing Manipulation Table	■ 'Pattern' ■ 'Replace-With'
Call Setup Rules Table	■ 'Condition' ■ 'Action Value'

String Examples

The following table provides configuration examples for using strings.

Table 3-2: Examples of Using Strings

Message Type	Condition	Action Subject	Action Type	Action Value	Description
`invite.request`	`header.user-agent contains 'X-Lite'`	`header.user-agent`	**Modify**	`'anonymous UA'`	If the INVITE's User-Agent header contains "X-Lite", replace it with "anonymous UA".
`invite.request`	`header.from.url.user=='101;ext=7166'`	`header.user-agent`	**Modify**	`'anonymous UA'`	If the INVITE's From header has the user part as "101;ext=7166", replace it with "anonymous UA".
`invite.request`	`body.sdp contains 'a=bbb\\a=ccc'`	`header.user-agent`	**Modify**	`'anonymous UA'`	If the INVITE's SDP contains these two lines: `a=bbb`

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					a=ccc then change the User-Agent header value to "anonymous UA"

Headers

This section describes the syntax used for SIP headers in the Message Manipulations table.

Syntax:

```
header.<header-name>.<header-index>.<sub-type>
```

where:

- <header-name> specifies the header name as it arrives in the message. For example: From, To, Contact (not case sensitive).
- <header-index> refers to a specific header, in the event where more than one header of the same type is present in the message (multiple header fields). The index starts at 0, therefore in order to refer to the first header in the list, the header-index value should be 0. For example, header.contact.2 would refer to the third header in the list.
  If <header-index> is not specified; however, a <sub-type> exists, then the sub-type would reference the first header in the list, i.e. header.contact.url.user is identical to header.contact.0.url.user.

  If both <header-index> and <sub-type> are not specified, then the subject would refer to all headers of this type. For example, to remove or modify all headers of a specific type, refer to the header as header.contact.
- <sub-type> specifies a specific part of the message. For example, url.user, url.host etc.

The SIP Group Name (IPGroup_SIPGroupName) parameter of the IP Groups table overrides inbound message manipulation rules that manipulate the host name in Request-URI, To, and/or From SIP headers. If you configure a SIP Group Name for an IP Group and you want to manipulate the host name in these SIP headers, you must apply your manipulation rule (Manipulation Set ID) to the IP Group as an Outbound Message Manipulation Set (IPGroup_OutboundManSet), when the IP Group is the destination of the call. If you apply the Manipulation Set as an Inbound Message Manipulation Set (IPGroup_InboundManSet), when the IP Group is the source of the call, the manipulation rule is overridden by the SIP Group Name.

Detailed Header Syntax

The following table describes the syntax to manipulate the various SIP headers:

Table 3-3: Syntax for Manipulating SIP Headers

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
Accept	Header itself	`header.accept`	-
Accept-Language	Header itself	`header.accept-language`	-
Allow	Header itself	`header.allow`	-
Authorization	Header itself	`header.authorization`	
Call-Id	Header itself	`header.call-id`	-
	Specific ID	`header.call-id.id`	-
Contact	Header itself	`header.contact`	-
	Expires	`header.contact.expires`	-
	Globally Routable UA URI	`header.contact.gruucontact`	-

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	(GRUU) contact		
	Enable GRUU	`header.contact.isgruu`	-
	Name	`header.contact.name`	-
	Parameter	`header.contact.param`	-
	Multiple Contact header fields	`header.contact.<number of fields>`	`header.contact.3`
	URL	`header.contact.url.<url>, where <url> can be:`	-
		■ type: Defines the type of URL: ✓ 1: Indicates a SIP URI (sip:) ✓ 2: Indicates a SIP Tel URI (tel:) ✓ 3: Indicates a fax URI (fax:) ✓ 4: Indicates a SIPS URI (sips:)	`header.contact.url.type == '1'`
		■ host: Indicates host part. The host by itself includes both domain name/IP address and port, e.g., 10.33.2.6:5070. However, you can indicate only the name/IP address or only the port: ✓ name: Indicates the host name ✓ port: Indicates the port	`header.contact.url.host.port`
		■ mhost: Indicates the SIP 'maddr' parameter (see RFC 3261)	–

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		■ userphone: Indicates the SIP 'user=phone' parameter (the tel URI). (See note below.)	`header.contact.url.userphone`
		■ looseroute: Indicates loose routing parameter ('lr') according to the Record-Route set (see note below)	-
		■ user: Indicates the user part of the URI (string)	`header.contact.url.user=='401'`
		■ transporttype: ✓ 0: UDP ✓ 1: TCP ✓ 2: TLS ✓ 3: SCTP	`header.contact.url.transporttype == '0'`
		■ param: Indicates a SIP parameter for the URI (can add, for example)	`header.contact.url.param.subject`
		Note: ■ For type, host, mhost, userphone, looseroute, user, and transporttype, the 'Action Type' field must be set to Modify. ■ For userphone and looseroute, configure the rule with the 'Action Value' field set to '0' (to remove) or '1' (to add).	-
Content-Disposition	Header itself	`header.content-disposition`	-
	Type	`header.content-disposition.type`	-
	Handling	`header.content-`	-

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		`disposition.handling`	
Content-Length	Header itself	`header.content-length`	-
Content-Type	Header itself	`header.content-type`	-
	Type	`header.content-type.type`	-
	Param	`header.content-type.param`	-
	Boundary	`header.content-type.boundary`	-
Cseq	Header itself	`header.cseq`	-
	Number	`header.cseq.num`	`header.cseq.num=='1'`
	Type	`header.cseq.type`	-
Date	Header itself	`header.date`	-
Diversion	Header itself	`header.diversion`	-
	Name	`header.diversion.name`	-
	Parameter	`header.diversion.param`	-
	Privacy - 1 (full) / 2 (off)	`header.diversion.privacy`	`header.diversion.privacy=='1'`
	Reason (enum)	`header.diversion.reason`	-
	Screen –	`header.diversion.screen`	-

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	yes / no		
	URL (see URL on page 13 for Contact header)	`header.diversion.url`	-
Event	Header itself	`header.event`	-

	Event Key ID
	Event package
		`header.event.eventkey`
		`header.event.eventkey.id`
		`header.event.eventkey.eventpackage`
			-

	Parameter	`header.event.param`	`header.event.param.itsp-abc`
Expires	Header itself	`header.expires`	-
	Expiry time	`header.expires.time`	-
From	Header itself	`header.from`	
	Name	`header.from.name`	
	Remove quotation marks surrounding display name	`header.from.quotecontrol` The 'Action Value' field must be set to '0'.	
	Parameter	`header.from.param`	`header.from.param.p1`
	Tag	`header.from.tag`	
	URL (see	`header.from.url`	`header.from.url.use`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	URL on page 13 for Contact header)		`r != '654'`
History-Info	Header itself	`header.history-info`	
Join	Header itself	`header.join`	
Max-Forwards	Header itself	`header.max-forwards`	
	Value	`header.max-forwards.val`	
Min-Se and Min-Expires	Header itself	`header.min-se` `header.min-expires`	
	Parameter	`header.min-expires.param`	
	Time	`header.min-expires.time`	
P-Asserted-Identity	Header itself	`header.p-asserted-identity`	
	Name (string)	`header.p-asserted-identity.name`	
	URL (see URL on page 13 for Contact header)	`header.p-asserted-identity.url`	`header.p-asserted-identity.url.host`
P-Associated-URI	Header itself	`header.p-associated-uri`	
	Name (string)	`header.p-associated-uri.name`	
	Parameter	`header.p-associated-`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		`uri.param`	
	URL (see URL on page 13 for Contact header)	`header.p-associated-uri.url`	
P-Called-Party-ID	Header itself	`header.p-called-party-id`	
	Name (string	`header.p-called-party-id.name`	
	Parameter	`header.p-called-party-id.param`	`header.p-called-party-id.param.p1`
	URL (see URL on page 13 for Contact header)	`header.p-called-party-id.url`	
P-Charging-Vector	Header itself	`header.p-charging-vector`	
P-Charge-Info	Header itself	`header.p-charge-info`	
P-Preferred-Identity	Header itself	`header.p-preferred-identity`	
	Name (string)	`header.p-preferred-identity.name`	
	URL (see URL on page 13 for Contact header)	`header.p-preferred-identity.url`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
Priority	Header itself	`header.priority`	
Privacy	Header itself	`header.privacy`	
	Privacy types	`header.privacy.privacy.<type>` where <type> can be: ■ none ■ header ■ session ■ user ■ critical ■ identity ■ history	`header.privacy.privacy.user`
Proxy-Authenticate	Header itself	`header.proxy-authenticate`	
Proxy-Authorization	Header itself	`header.proxy-authorization`	
Proxy-Require	Header itself	`header.proxy-require`	
	SIP Capabilities	`header.proxy-require.capabilities.<capability>` where <capability> can be: ■ earlymedia ■ reliableresponse ■ timer ■ earlysession	`header.proxy-require.capabilities.earlymedi`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		■ privacy ■ replaces ■ history ■ unknown ■ gruu ■ resourcepriority ■ targetdialog ■ sdpanat	
RAck	Header itself	`header.rack`	
Reason	Header itself	`header.reason`	
	Reason types	`header.reason.reason.<type>` where <type> can be: ■ reason ■ cause ■ text	`header.reason.reason.reason`
	MLPP: Type: Preemption (0), MLPP (1) cause	`header.reason.mlpp`	
Record-Route	Header itself	`header.record-route`	
Referred-By	Header itself	`header.referred-by`	
	Parameter	`header.referred-`	`header.referred-`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		`by.param`	`by.param.p1`
	URL (see URL on page 13 for Contact header)	`header.referred-by.url`	`header.referred-by.url.host`
	Display Name	`header.referred-by.name`	`header.referred-by.name == 'user1'`
	Original Referred-By header value	`header.referred-by.original`	`header.referred-by.original == 'sip:referrer@ref.example;cid=X'`
Refer-To	Header itself	`header.refer-to`	
	The From tag of the call on the device being replaced; it is part of the value of the Replaces URI header, in the Refer-To header of an outgoing request	`header.refer-to.fromtag`	`header.refer-to.fromtag == 'some_tag_value'`
	The To tag of the call on the device	`header.refer-to.totag`	`header.refer-to.totag == 'some_tag_value'`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	being replaced; it is part of the value of the Replaces URI header, in the Refer-To header of an outgoing request		
	Represents XRawDataInfoHeader as a URI header, when the header is in an outgoing Refer request. **Note:** Applicable only to the gateway application.	`header.refer-to.addparams`	`header.refer-to.addparams len> 0`
	Display Name	`header.refer-to.name`	`header.refer-to.name == 'user1'`
	Parameter	`header.refer-to.param`	`header.refer-to.param.p1`
	A Boolean value indicating	`header.refer-to.isreplacesused`	`header.refer-to.isreplacesused == 1`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	whether the Refer-To header in the outgoing request has URI header Replaces with call identifiers for a call on the device		
	The Call-ID of the call on the device being replaced; it is part of the value of the Replaces URI header, in the Refer-To header of an outgoing request	`header.refer-to.replacedcallid`	`header.refer-to.replacedCallID len> 0`
	URL (see URL on page 13 for Contact header)	`header.refer-to.url`	`header.refer-to.url.host`
Remote-	Header itself	`header.remote-party-id`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
Party-ID			

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	Counter	`header.remote-party-id.counter`	
	Name	`header.remote-party-id.name`	
	Number Plan	`header.remote-party-id.numberplan` where <numberplan> can have the following value: ■ 1: ISDN ■ 3: Data ■ 4: Telex ■ 8: National ■ 9: Private ■ 15: Reserved	
	Number Type	`header.remote-party-id.numbertype`	
	Parameter	`header.remote-party-id.param`	
	Privacy (see Privacy types on page 19 header for description)	`header.remote-party-id.privacy`	
	Reason types	`header.remote-party-id.reason` where reason can equal the following enumeration value: ■ 1: busy ■ 2: immediate	`header.remote-party-id.reason=='1'`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		■ 3: no answer	
	Screen – Yes / No	`header.remote-party-id.screen`	
	Screen Indicator types	`header.remote-party-id.screening` where screening can equal the following enumeration value: ■ -1: Screening not included ■ 0: user provided ■ 1: user passed ■ 2: user failed ■ 3: network provided	`header.remote-party-id.screening == 0`
	URL (see URL on page 13 for Contact header)	`header.remote-party-id.url`	
Replaces	Header itself	`header.replaces`	
Request-URI	Header itself	`header.request-uri`	
	Method	`header.request-uri.method`	
	Method Type	`header.request-uri.methodtype` The following enumerations are used to represent the SIP methods: ■ 5: INVITE ■ 7: BYE ■ 8: OPTIONS	`header.request-uri.methodtype == '5'` (i.e., SIP method is INVITE message)

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		■ 9: ACK ■ 10: CANCEL ■ 11: REGISTER ■ 12: INFO ■ 13: MESSAGE ■ 14: NOTIFY ■ 15: REFER ■ 16: SUBSCRIBE ■ 17: PRACK ■ 18: UPDATE ■ 19: PUBLISH ■ 21: SERVICE	
	URI	`header.request-uri.uri`	
	URL (see URL on page 13 for Contact header)	`header.request-uri.url`	`header.request-uri.url.user == '101'`
Require	Header itself	`header.require`	
	SIP Capabilities (see SIP Capabilities on page 19 for Proxy-Require header)	`header.require`	`header.require.earlymedia`
Resource-Priorit	Header itself	`header.resource-priority`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
y			
	Namespace	header.resource-priority.namespace	
	RPriority	header.resource-priority.rpriority	
Retry-After	Header itself	header.retry-after	
	Time	header.retry-after.time	
RSEQ	Header itself	header.rseq	
Server or User-Agent	Header itself	header.user-agent header.server	
Service-Route	Header itself	header.service-route	
	Service route list entry	header.service-route.<entry>.serviceroute	header.serviceroute.1.serviceroute
Session-Expires	Header itself	header.session-expires	
	Parameter	header.session-expires.param	header.session-expires.param.longtimer
	Refresher	header.session-expires.refresher	**Note:** The Action Value '1' sets it to "UAC"; the value '2' sets it to "UAS" (i.e., UA type doing the refreshing)
	Time	header.session-expires.time	
SIP	Header	header.sip-etag	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
ETag	itself		
SIP If-Match	Header itself	`header.sip-if-match`	
Subject	Header itself	`header.subject`	
	Subject	`header.subject.subject`	
Subscription State	Header itself	`header.subscription-state`	
Supported	Header itself	`header.supported`	
	SIP Capabilities (see SIP Capabilities on page 19 for Proxy-Require header)	`header.supported.capabilities.<capability>`	`header.supported.capabilities.path`
Target Dialog	Header itself	`header.target-dialog`	
To	Header itself	`header.to`	
	Display name	`header.to.name`	
	Parameter	`header.to.param`	`header.to.param.artist`
	tag	`header.to.tag`	
	URL (see URL on	`header.to.url`	`header.to.url.userphone`

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
	page 13 for Contact header)		
Unknown headers	Header itself	`header.<unknown header name>`	`header.color`
Unsupported	Header itself	`header.unsupported`	
	SIP Capabilities (see SIP Capabilities on page 19 for Proxy-Require header)	`header.unsupported.capabilities.<capability>`	`header.unsupported.capabilities.path`
User-To-User and X-UserToUser	Header itself	`header.user-to-user` `header.x-usertouser`	
	User-to-User Descriptor	`header.user-to-user.user2user` `header.x-usertouser.user2user`	
	Protocol Descriptor (PD)	`header.user-to-user.pd` `header.x-usertouser.pd`	
	Data Type (enum)	`header.user-to-user.datatype` `header.x-usertouser.datatype`	`header.usertouser.datatype == '1'`
	Parameter	`header.user-to-user.param` `header.x-`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		`usertouser.param`	
Via	Header itself	`header.via`	
	Alias	`header.via.alias`	
	Branch	`header.via.branch`	
	Host name	`header.via.host`	
	Via parameter 'maddr'	`header.via.maddrip`	
	Parameter	`header.via.param`	
	Port	`header.via.port` Where port can have one of the following values: ■ -1: No rport parameter ■ 0: The rport parameter is without a value ■ 1-65535: The rport parameter with the specified value **Note:** "port" refers to the Via header's rport parameter.	`header.via.port=='0'`
	Transport type	`header.via.transporttype` Where transporttype can have one of the following values: ■ 0: UDP ■ 1: TCP ■ 2: TLS ■ 3: SCTP	`header.via.0.transporttype == '0'`
Warning	Header itself	`header.warning`	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
WWW Authenticate	Header itself	`header.www-authenticate`	
X-AC-Action	Header itself	`header.x-ac-action`	
X-Channel	Header itself	`header.x-channel`	
	TrunkID	`header.x-channel.trunkid`	
	BChannel	`header.x-channel.bchannel`	
	BoardIP	`header.x-channel.boardip`	
	HeaderType	`header.x-channel.headertype`	
X-RTP-Stat	Header itself	`header.x-rtp-stat`	

Header Examples

The following table provides examples of syntax for indicating header fields.

Table 3-4: Header Field Syntax Examples

Header	Description
`header.to`	Defines the top level of the To header.
`header.to.url.user`	Defines the user part in the header SIP URL.
`header.from.url.host`	Defines the host part in the From header.
`header.from.name`	Defines the display name in the From header.

Header	Description
header.newheader	Defines a header newheader.
header.contact.param.newparam	Defines the parameter newparam of a Contact header.
header.refer-to.url.host	Defines the host part of the Refer-To header.
header.diversion.reason	Defines the Reason parameter in the Diversion header.
header.supported.capabilities.path	Defines the supported headers capabilities path.
header.supported.capabilities.replaces	Defines the supported headers capabilities replaces.
header.max-forwards.val	Defines the value of the Max-Forwards header.
header.request-uri.methodtype	Defines the method in the Request-URI.
header.remote-party-id.0.partytype	Defines the party type in the first Remote-Party-ID header.
header.contact.3	Defines the third Contact header.
header.via.2.url.user	Defines the user part of the second Via header.

The following table provides examples of manipulation rules for headers.

Figure 3-1: Header Field Manipulation Rules Examples

Message Type	Condition	Action Subject	Action Type	Action Value	Explanation
register.	header.from.url.user == '101' OR	header.from.url.user	**Modify**	'2000'	If the user part of the From

Message Type	Condition	Action Subject	Action Type	Action Value	Explanation
request	header.from.url.user == '1000'				header's URL in the REGISTER request is "100" or 1000", change it to "2000".
register		header.to.url.host.name	**Modify**	'audiocodes.com'	In the REGISTER message, change the host part of the To header's URL to "audiocodes.com".
invite		header.from.name	**Modify**	header.contact.url.user	In the INVITE message, change the display name of the From header to the user part of the Contact header's URL.
invi		header.newhe	**Add**	'informa	In the

Message Type	Condition	Action Subject	Action Type	Action Value	Explanation
te.request		ader		tion to client'	INVITE request, add a header called "newheader" with the value "'information to client'".
subscribe	header.via.transporttype=='1'	header.to.param.transporttype	**Add**	'TCP'	If the transport type of the Via header in the SUBSCRIBE message is "1" (TCP), add the transport type "TCP" to the To header.
invite	header.allow !contains 'REFER'	header.allow	**Modify**	header.allow +',REFER'	If the Allow header in the INVITE message doesn't include the value "REFER",

Message Type	Condition	Action Subject	Action Type	Action Value	Explanation
					add "REFER" in the outgoing INVITE.

Body

This section describes the syntax used for the SIP body in the Message Manipulations table.

The syntax can also be used to manipulate (add, modify, or remove) any header preceding a body part in a multipart body of a SIP message. Up to three headers can be manipulated per body part.

Syntax:

- Manipulation of the body itself:

```
body.<body name>
```

- Manipulation of a specific header preceding the body part:

```
body.<body part name - from Content-Type header>.header.<header to manipulate>
```

When manipulating the Content-Type and Content-Disposition headers, the '.header' key may be omitted from the syntax.

The <body name> specifies the body name as it appears in the received message. For example, 'application/sdp' (case-insensitive), or 'application/pidf+xml' as shown below for a multipart body:

```
...
--boundary_ac186cboundary_ac166e
Content-Type: application/pidf+xml
Content-Disposition: render;handling=required
Content-ID: < example@example.edu >
<?xml version="1.0" encoding="UTF-8"?><presence
xmlns="urn:ietf:params:xml:ns:pidf" entity="example@example.edu "><tuple
```

```
id="0"><status><geopriv
xmlns="urn:ietf:params:xml:ns:pidf:geopriv10"><location-info><Point
srsName="urn:ogc:def:zzz:EPSG::4326" xmlns="http://www.opengis.net
/gml"><pos>26.07686 -80.25351</pos></Point><civicAddress
xmlns="urn:ietf:params:xml:ns:pidf:geopriv5:civicAddr"><country>US</country><
A1>FL</A1><A2></A2><A3>SUE</A3><PRD></PRD><RD>NW 4TH
ST</RD><STS></STS><POD></POD><HNO>8000</HNO><HNS></HNS><LO
C></LOC><NAM>EXAMPLE
UNIVERSITY</NAM><PC>33328</PC><ELIN>Teams_
NSUSOC5</ELIN></civicAddress></location-info><usage-rules><retransmission-
allowed
xmlns="urn:ietf:params:xml:ns:pidf:geopriv5:basicPolicy">true</retransmission-
allowed></usage-
rules><method>LIS</method></geopriv></status></tuple></presence>
--
```

Body Examples

The following table provides examples of the syntax for indicating the SIP message body.

Table 3-5: Message Body Syntax Examples

Subject	Description
`body.application/x-nt-mcdn-frag-hex`	Adds or removes this 'unknown' body type.
`body.sdp`	Defines the SDP in the body.
`body.application/pidf+xml.header.Content-ID`	Adds, modifies or removes the Content-ID header.

The following table provides configuration examples of manipulation rules for the message body.

Table 3-6: Message Body Manipulation Rules Examples

Message Type	Condition	Action Subject	Action Type	Action Value
invite	body.sdp !exists	body.application/ x-nt-mcdn-frag-hex	Add	'a=0981233\\b=12rewwer\\note=newlinechar acter'
invite	header.request-uri.url.user contains '+1811' AND Header.Priority !exists AND Header.from.URL.User contains '+1732'	body.application/pidf+xml	Add	'<?xml version=\"1.0\" encoding=\"UTF-8\"?><presence xmlns=\"urn:ietf:params:xml:ns:pidf\" entity=\"example@example.edu\"><tuple id=\"0\"><status><geopriv xmlns=\"urn:ietf:params:xml:ns:pidf:geopriv5\"><location-info>'
invite	body.application/pidf+xml exists	body.application/pidf+xml.Content-Disposition	Modify	'render;handling=required'
invite	body.application/pidf+xml exists	body.application/pidf+xml.header.Content-ID	Add	'<example@example.edu>'
invite.request		body.mwi	Add	'Messages-Waiting: yes\\Message-Account: sip:alice@vmail.example.com\\Voice-Message:

Message Type	Condition	Action Subject	Action Type	Action Value
				2/8 (0/2)'
any		body.mwi.summary.newmsgs	Modify	'23'
invite		body.mwi.summary.oldmsgs	Modify	'18'
invite		body.mwi.summary.newurgentmsgs	Modify	'12'
any		body.mwi.summary.oldurgentmsgs	Modify	'67'
invite		body.mwi.pending	Modify	'8'
invite		body.mwi.messagewaiting	Modify	'2'

Parameters

This section describes the syntax used for the following SIP parameter types in the Message Manipulations table:

- Message parameters (see Message Parameter Syntax on the next page)

- IP Group parameters (see IP Groups Table Parameter Syntax on page 43)
- Call parameters (see Call Parameter Syntax on page 46)
- Payphone parameters (see Payphone Parameter Syntax on page 47)
- Global session ID parameters (see Global Session ID Parameter Syntax on page 49)
- Call trigger parameters (see Call Trigger Parameter Syntax on page 50)

Message Parameter Syntax

The following table describes the syntax used for SIP message parameters.

Table 3-7: Message Parameter Syntax

Subject	Description
`param.message.sdp.ip`	Specifies the address in the SDP. **Note:** The parameter can be used for read-write operations in all message-syntax based tables.
`param.message.sdp.originusername`	Specifies the username in the Origin ('o=') field of the SDP. **Note:** The parameter can be used for read-write operations in all message-syntax based tables.
`param.message.sdp.ip-for-url`	Specifies the address in the SDP. IPv6 addresses are enclosed in square brackets while IPv4 addresses appear without. **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
`param.message.sdp.rtpmode`	Specifies the RTP mode in the SDP. **Note:** The parameter can be used for read-write operations in all message-syntax based tables.
`param.message.sdp.originaddress`	Specifies the origin address in the SDP. **Note:** The parameter can be used for read-write operations in all message-syntax based tables.
`param.message.sdp.port`	Specifies the port in the SDP. **Note:** The parameter can be used for read-write operations in all message-syntax

Subject	Description
	based tables.
`message.incoming.remote-port`	Specifies the remote (peer) port for the source of the message, as a string. **Note:** ■ The parameter replaces the old "param.message.address.src.port" parameter. ■ The parameter can only be used for read-only operations in the message-syntax based tables.
`message.outgoing.remote-port`	Specifies the remote (peer) port for the destination of the message, as a string. **Note:** ■ The parameter replaces the old "param.message.address.dst.port" parameter. ■ The parameter can only be used for read-only operations in the message-syntax based tables.
`message.incoming.local-port`	Specifies the local port for the source of the message, as a string (port on which the message is received). **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
`message.outgoing.local-port`	Specifies the local port for the destination of the message, as a string (port from which the message is sent). **Note:** ■ The parameter can be used for write operations in the Call Setup Rules table and read-only operations in the other message-syntax based tables. ■ It has a non-zero value for the relevant message only after being modified by Call Setup Rules for that message.

Subject	Description
param.message.address.src.ip	Specifies the IP address as a string for the source of the message. The IP address is returned as is. **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.dst.ip	Specifies the IP address as a string for the destination of the message. The IP address is returned as is. **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.src.ip-for-url	Specifies the IP address as a string for the source of the message. IPv6 addresses are enclosed in square brackets while IPv4 addresses appear without. An example of a returned IPv6 address is [2620:0:2ef0:7070:250:60ff:fe03:32b7]. This is useful for creating a SIP URI, which would look like "sip:6000@[2620:0:2ef0:7070:250:60ff:fe03:32b7]:5060;transport=tcp" **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.dst.ip-for-url	Specifies the IP address as a string for the destination of the message. IPv6 addresses are enclosed in square brackets while IPv4 addresses appear without. An example of a returned IPv6 address is [2620:0:2ef0:7070:250:60ff:fe03:32b7]. This is useful for creating a SIP URI, which would look like "sip:6000@[2620:0:2ef0:7070:250:60ff:fe03:32b7]:5060;transport=tcp" **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.src.<tr	Specifies the transport type as a string for

Subject	Description
ansporttype>	the source of the message, where <transporttype> can be one of the following: ■ UDP ■ TCP ■ TLS **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.dst.<transporttype>	Specifies the transport type as a string for the destination of the message, where <transporttype> can be one of the following: ■ UDP ■ TCP ■ TLS **Note:** The parameter can only be used for read-only operations in the message-syntax based tables.
param.message.address.src.sipinterface	Specifies the SIP Interface ID on which the message is received (source). **Note:** The parameter can only be used for read-only operations ('Action Value' and 'Condition' fields only) in the message-syntax based tables.
param.message.address.dst.sipinterface	Specifies the SIP Interface ID on which the message is sent (destination). **Note:** The parameter can only be used for read-only operations ('Action Value' and 'Condition' fields only) in the message-syntax based tables.

IP Groups Table Parameter Syntax

This section describes the syntax for representing parameters in the IP Groups table.

Syntax:

```
param.ipg.src|dst|<ID>|<Name>.host|id|is-alive|name|tags|type| user|user-defined
```

Where:

- *ID*: Specifies the IP Group by table row index.
- *Name*: Specifies the IP Group by name.
- *ipg.src|dst*: These are determined according to the context in which they are used:
  - **Message Manipulations table:**
    - **Pre-Classification:** The *param.ipg.src|dst* parameters are not relevant. For rules of a Manipulation Set that are assigned in the SIP Interfaces table ('Pre-classification Manipulation Set ID' parameter), use *param.ipg.<Name>|<ID>*.
    - **Inbound Manipulation Set:** You can use *param.ipg.src|dst* and *param.ipg.<Name>|<ID>* because these rules are run after both Classification and Routing stages complete. For a SIP message (request or response) that is received by the device from one UA and then sent to another UA, *param.ipg.src* is the IP Group to which the message is sent; *param.ipg.dst* is the IP Group from where the message was received.
    - **Outbound Manipulation Set:** You can use *param.ipg.src|dst and ipg.<Name>|<ID>* because these rules are run after both Classification and Routing stages complete. For a SIP message (request or response) that is received by the device from one UA and then sent to another UA, *param.ipg.src* is the IP Group from where the message was received; *param.ipg.dst* is the IP Group to where the message is sent.
  - **Call Setup Rules:** The *param.ipg.dst* parameter is accessible only when the Call Setup Rule is invoked from the IP-to-IP Routing table. The *param.ipg.src* parameter may be used when the Call Setup Rule is run after the Classification stage (i.e., when the Call Setup Rule is run from the IP Groups table or the IP-to-IP Routing table). This parameter can't be used when Call Setup Rules are run from the SIP Interfaces table.
  - **Message Conditions:** The *param.ipg.src|dst* parameters are not relevant. Use only *param.ipg.<Name >|<ID>* because message conditions are run before the Classification or Routing stages complete.

When using the *param.ipg.<Name>*, the IP Group name is case-sensitive and cannot contain spaces or dots (.).

The specific parameters - - fourth- level in the syntax above - - (host|id|is-alive|name|tags|type|user|user-defined) are described in the following table:

Table 3-8: IP Group Parameter Syntax

IP Group Parameter	Description
'SIP Group Name'	
`host`	Specifies the IP Group's host name.
'Index '	
`id`	Specifies the table row index (ID) of the given IP Group. For example, the following checks if the index of the source IP Group is 5: `param.ipg.src.id=='5'`
'Keep-Alive'	
`is-alive`	Specifies the IP Group's connectivity status, using the keywords 'true' or 'false' (online or offline, respectively). For example, the following specifies a condition checking if IP Group at index 5 is online: `param.ipg.5.is-alive=='true'` **Note:** ■ User-type IP Groups are always online. ■ A Gateway-type IP Group is online when it is registered with the device. ■ A Server-type IP Group is online when its' associated Proxy Set has at least one valid IP address and either has keep-alive disabled or has keep-alive enabled and keep-alive transactions with the proxy server(s) are successful
'Name'	
`name`	Specifies the name of the given IP Group. For example, the following condition checks if the name of the source IP Group is "ITSP-Site1": `param.ipg.src.name=='ITSP-Site1'`
'Tags'	
`tags.<Tag Name>`	Specifies the value of an IP Group's tag. By not specifying a tag name, you can refer to all the tags and their values associated with the IP Group. For example, the following specifies the value of tag, "city"

IP Group Parameter	Description
	for source IP Groups: `param.ipg.src.tags.city`
'Type'	
`type`	Specifies the IP Group's type (server, user, and gateway). For example, the following condition checks if the source IP Group is of type server: `param.ipg.src.type=='server'`
'Contact User'	
`user`	Specifies the IP Group's user part of the From, To, and Contact headers of SIP REGISTER messages, and the user part of the Contact header of INVITE messages (in other words, it specifies the value of the 'Contact User' parameter for the IP Group). For example, the following condition checks if the source IP Group has Contact "jdoe": `param.ipg.src.user=='jdoe'`
'Message Manipulation User-Defined String 1 / 2'	
`user-defined.0|1`	Specifies the IP Group's user-defined string for manipulation rules in the IP Group table, where: ■ 0 uses the string configured for the 'Message Manipulation User-Defined String 1' parameter in the IP Group table ■ 1 uses the string configured for the 'Message Manipulation User-Defined String 2' parameter in the IP Group table For example, the following specifies the 'Message Manipulation User-Defined String 2' parameter of the source IP Group: `param.ipg.src.user-defined.1`

Call Parameter Syntax

The following table describes the syntax used for Call parameters in the Message Manipulations table.

Table 3-9: Call Parameter Syntax

Subject	Description
`param.call.<src/dst>.user`	Specifies the source or destination username during run-time.
`param.call.<src/dst>.nat`	Enables manipulation of a SIP message depending on whether (=='true') or not (=='false') the source or destination of the message is located behind NAT. The keywords can be used in the 'Condition' or 'Action Value' parameters in the Message Manipulations table. Message Manipulation rules using the keywords are applicable only to message manipulation on the outbound leg (i.e., the rules can only be assigned to the 'Outbound Message Manipulation Set' parameter in the IP Group table.

Payphone Parameter Syntax

The syntax for indicating payphone calls can be used in the Call Setup Rules table ('Condition' and 'Action Subject' fields).

Syntax:

```
param.payphone==<'1' or '0'>
```

The following table provides examples:

Table 3-10: Payphone Parameter Example

Request Type	Request Key	Attributes To Get	Condition	Action Subject	Action Type	Action Value	Description
			`Param.payphone=='1'`		**Exit**	`True`	If the call is a pay

Request Type	Request Key	Attributes To Get	Condition	Action Subject	Action Type	Action Value	Description
							phone, then exit the CSR.
LDAP	'telephonenumber='+Param.Call.Src.User	telephoneNumber	LDAP.Attr.telephoneNumber exists	Param.payphone	**Modify**	'1'	If the above is false, then query the LDAP server for the callers number and if exis

Request Type	Request Key	Attributes To Get	Condition	Action Subject	Action Type	Action Value	Description
							ts, then change the payphone parameter to "1".

Global Session ID Parameter Syntax

The syntax for indicating the global session ID can be used in the Call Setup Rules table and Message Manipulations table ('Action Value' field). This is used for obtaining the global session ID.

Syntax:

```
Param.Session.GID
```

The following table provides examples:

Table 3-11: Global Session ID Parameter Example

Action Subject	Action Type	Action Value	Description
`header.GID`	**Add**	`param.session.gid`	Global Session ID value is added to a new header called "GID".

Call Trigger Parameter Syntax

The syntax for indicating the call trigger can be used in the Call Setup Rules table ('Action Value' field). This is used for obtaining the trigger (e.g., due to a SIP 3xx) for re-routing calls to alternative destinations. This applies to call re-routing that is handled locally by the device (e.g., 'Remote REFER Mode' configured to Handle Locally for call transfer). The IP-to-IP Routing table's parameter 'Call Trigger' defines the condition required for re-routing the call.

Syntax:

```
Param.Routing.Trigger
```

The following table provides examples:

Table 3-12: Global Session ID Parameter Example

Action Subject	Action Type	Action Value	Description
`header.Trigger`	**Add**	`Param.Routing.Trigger`	Call trigger value added to a new header called "Trigger".

Parameter Examples

The following table provides configuration examples using parameters.

Table 3-13: Parameter Examples

Message Type	Condition	Action Subject	Action Type	Action Value	Description
		`header.contact.url.ac-int`	**Modify**	`param.message.address.src.sipinterface`	Adds the ID number of the SIP Interface on which the message is received, to

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					the value of the "ac-int" parameter in the URL of the Contact header.
	param.message.sdp.address == '10.132.10.101'	header.IPSource	Add	param.ipg.src.id	If the address in the SDP is 10.132.10.101, the SIP header "IPSource" is added and set to the value of the Index of the source IP Group.
invite.response.200	param.message.sdp.rtpmode=='inactive'	header.origin	Add	param.message.sdp.originaddress	In 200 OK messages, if the RTP mode is inactive, add a new header, "origin" whose value is set to the address in the origin ('o=') SDP
	param.message.sdp.rtpmode==	header.from.param.origin	Add	param.message.sdp.originaddress	If the RTP mode is inactive, add a

Message Type	Condition	Action Subject	Action Type	Action Value	Description
	'inactive'				new parameter, "origin" to the From header. The value of the parameter is set to the 'o=' address in the SDP.
subscribe.request		header.to.param.user	Add	param.call.src.user	In SUBSCRIBE messages, add the parameter, "user" to the To header. The value is set to the source username.
invite.response		header.request-uri.url.param.myname	Add	param.ipg.src.host	For INVITE responses, adds the "myname" parameter to the Request-URI. The parameter value is taken from the 'SIP Group Name' field of the source IP Group.

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite		header.MyCustomHeader	Add	param.ipg.dst.user-defined.0	For INVITE messages, add a header called "MyCustomHeader" and whose value is taken from the IPGroup_MsgManUserDef1 field of the destination IP Group.
any.request		header.session-expires.refresher	Modify	'1'	Manipulates the 'refresher' parameter to "UAC" in the Session-Expires header (i.e., UAC is doing the refreshing). For example: Session-Expires: 180;refresher=uac
invite	param.message.sdp.rtpmode=='sendonly' and	param.message.sdp.rtpmode	Modif	'sendrecv'	If the device determines that the

Message Type	Condition	Action Subject	Action Type	Action Value	Description
	param.call.dst.nat=='true'		y		destination of the INVITE message is located behind NAT (param.call.dst.nat=='true'), and the RTP mode in the SDP of the incoming INVITE is 'sendonly' (param.message.sdp.rtpmode=='sendonly'), it changes the RTP mode to 'sendrecv' in the SDP of the outgoing INVITE.
invite	param.ipg.src.tags.city exists	header.City	Add	param.ipg.src.tags.city	For INVITE messages, if the source IP Group has the tag "city", add a header called "City" and set its value to the value of the tag "city".

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite	param.ipg.src.id=='34'	header.X-ID	Add	param.ipg.src.id	For INVITE messages, if the source IP Group ID is 34, add a header called "X-ID" and set its value to the ID of the source IP Group.
invite	param.ipg.src.type=='server'	header.X-Type	Add	param.ipg.src.type	For INVITE messages, if the source IP Group ID type is Server, add a header called "X-Type" and set its value to the Type (i.e., "Server") of the source IP Group.

Example for IP Group Keep-Alive

This example uses a Message Condition rule based on IP Group connectivity status (param.ipg.<Name>.is-alive) to determine the destination of the call. The device routes the calls between IP Groups "ITSP" and "MainServer". "MainServer" is always accessible to the device, but it in turn routes calls to the "Cloud" IP Group, which may not always be accessible(for whatever reason). If "Cloud" is inaccessible (offline), the device routes the call to an alternative server ("AltServer").

1. Proxy Sets table configuration:

Index	Name	Proxy Keep-Alive
1	ITSP	Disable
2	Cloud	Using OPTIONS
3	Main-Server	Disable
4	Alt-Server	Disable

2. IP Groups table configuration:

Index	Name	Proxy Set
1	ITSP	ITSP
2	Cloud	Cloud
3	Main-Server	Main-Server
4	Alt-Server	Alt-Server

3. Message Conditions table configuration:

Index	Name	Condition
1	Cloud-Status	param.ipg.Cloud.is-alive == 'true'

4. IP-to-IP Routing table configuration:

Index	Source IP Group	Alternative Route Options	Message Condition	Destination IP Group
1	ITSP	Route Row	Cloud-Status	Main-Server
2	ITSP	Route Row		Alt-Server

4 Advanced Manipulation Features

This chapter describes advanced features that you can use for manipulating SIP messages.

Wildcards for Header Removal

The device supports the use of the "*" wildcard character to remove headers. The "*" character may only appear at the end of a string. For example, "X-*" is a valid wildcard request, but "X-*ID" is not.

Below are examples of using the wildcard:

- header.p-*: Removes all headers that have the prefix "p-".
- header.x-vendor*: Removes all headers that start with "x-vendor".

The wildcard doesn't remove the following headers: Request-Uri, Via, From, To, Callid, Cseq, and Contact.

Random Characters

The following describes the syntax for random characters:

- To specify random letters in the range a to z in Message Manipulation rules:

  Syntax:

  ```
  rand.string.<n>.a.z
  ```

  where *n* is the number of random letters you wish to specify in the range a to z.

- To specify random letters and/or numeric characters in the range 0 to z in the Message Manipulations table:

  Syntax:

  ```
  Rand.string.<n>.0.z
  ```

  where *n* is the number of random letters and/or numeric characters you wish to specify in the range 0 to z.

- To specify a random number between n and m in the Message Manipulations table:

  Syntax:

  ```
  Rand.number.<n>.<m>
  ```

  where:

- *n* specifies the start value of the range of the random numbers that you wish to specify, in the range 0 to 2147482647.
- *m* specifies the end value of the range of the random numbers that you wish to specify, in the range 0 to 2147482647.

The following table provides configuration examples for using random letters and numeric characters in the Message Manipulations table.

Figure 4-1: Examples using Random Letters and Numeric Characters

Message Type	Action Subject	Action Type	Action Value	Comment
`invite.request`	`header.myrandom`	**Add**	`Rand.string.10.0.9`	String of 10 characters comprised of digits between 0 and 9.
`invite.request`	`header.myrandomString`	**Add**	`Rand.string.56.A.Z`	String of 56 characters comprised of letters between A and Z.
`invite.response`	`header.NumberaAndChars`	**Add**	`Rand.string.12.0.z`	String of 12 characters comprised of digits and letters between 0 and z (ASCII printable characters), for example, ebJ803GpK13@.
`invite.response.4xx`	`header.myrandomNmber`	**Add**	`Rand.number.50.100`	Single integer between 50 and 100.

SDP Body Fields

You can configure message manipulation based on the SDP body fields for read and/or write operations on the SDP. This can be done in the following tables and corresponding fields:

- **Message Manipulations table:** 'Condition', 'Action Subject', and 'Action Value' fields
- **Message Conditions table:** 'Condition' field
- **Call Setup Rules table:** 'Request Key', 'Condition', 'Action Subject', and 'Action Value' fields

Source IP Address

You can use the source IP address in the SDP body for message manipulation. For example, you can configure a manipulation rule to add a SIP Diversion header to incoming INVITE messages if the SDP contains a specific IP address, or a prefix or suffix of this IP address.

Syntax:

- Specific IP address:

```
param.message.sdp.ip
```

  For example:

```
param.message.sdp.ip=='10.33.37.78’
```

- IP address suffix:

```
param.message.sdp.ip suffix
```

  For example:

```
param.message.sdp.ip suffix '10.10'
```

- IP address prefix:

```
param.message.sdp.ip prefix
```

  For example:

```
param.message.sdp.ip prefix '10.132'
```

RTP Mode

You can use the RTP mode in the SDP body for message manipulation.

Syntax:

```
param.message.sdp.rtpmode
```

Possible values include the following:

- sendonly
- sendrecv
- inactive

For example:

```
param.message.sdp.rtpmode=='sendrecv'
```

Origin Username

You can use the username in the "o=" field of the SDP body for message manipulation.

Syntax:

```
param.message.sdp.originusername
```

Origin IP Address

You can use the IP address in the "o=" field of the SDP body for message manipulation.

Syntax:

```
param.message.sdp.originaddress
```

Possible values include any IP address.

Port

You can use the first audio active media port number (i.e., port number greater than 0) in the "m=" field of the SDP body for message manipulation.

Syntax:

```
sdp.port
```

IP Address

You can use the IP address of the first active media (port greater than 0) for message manipulation. The IP address is taken from the media "c=" field (the "c=" field below the "m=" field) of the SDP body. Note that if the "m=" field doesn't contain a "c=" field, the IP address is taken from the global "c=" field (the "c=" field at the top of the SDP).

Syntax:

```
sdp.address
```

SDP Examples

Below are manipulation examples using the SDP body:

Figure 4-2: Examples using SDP Body Fields

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite.request		header.custom-rtp-address	Add	param.message.sdp.ip	Copies the port and IP address in the SDP body to a customized SIP header (e.g., Custom-RTP-Address/Port) in the outgoing INVITE message.
invite.request		header.custom-rtp-port	Add	param.message.sdp.port	
reinvite.request	param.message.sdp.ip == '0.0.0.0'	param.message.sdp.rtpmode	Modify	'sendonly'	Changes the RTP mode to sendonly if the SDP "c=" field's address is 0.0.0.0.
		param.message.sdp.ip	Mod	param.message.sdp.originaddress	Changes the SDP "c=" field to the

Message Type	Condition	Action Subject	Action Type	Action Value	Description
			Modify		same address as the "o=" field.
invite	param.message.sdp.rtpmode=='sendrecv'	var.call.src.0	**Modify**	'1'	Uses the RTP mode as a condition.
invite.response.200	var.call.dst.0=='1'	param.message.sdp.rtpmode	**Modify**	'sendonly'	
invite	param.message.sdp.ip=='10.33.37.78'	header.diversion	**Add**	<sip:12345@p4.isp.com>;reason=no-answer	Adds a Diversion header ("Diversion: <sip:12345@p4.isp.com>;reason=no-answer") to incoming INVITE messages if the SDP contains the IP address 10.33.37.78 or the prefix of this IP address

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					(10.33). The IP address is contained in the "c=" field of the SDP (e.g., "c=IN IP4 10.33.37.75").
invite	param.message.sdp.ip prefix '10.33'	header.diversion	Add	<sip:12345@p4.isp.com>;reason=no-answer	
invite.request	body.sdp exists	Header.SDP-Origin-UserName	Add	param.message.sdp.originusername	Adds a customized header “SDP-Origin-UserName”, where the username is obtained from the “o=” field in the SDP body, if the INVITE message contains an SDP body.
invite.request	body.sdp exists	param.message.sdp.originusername	Modify	'myname'	Changes the username in the “o=” field in the SDP body to "myname", if the INVITE message

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					contains an SDP body:

Regular Expressions (Regex)

You can configure SIP header manipulation rules using regular expressions (regex). Regex is a special text string pattern matching engine which is used to define the condition that must exist in order to use a specific manipulation rule. If the SIP header matches the regex pattern, then the "action" of the manipulation rule is applied to the SIP message. Executing a regex pattern also creates sub-expressions. The sub-expressions are referenced using the $n syntax, where n is a digit in the range of 1 to 13 (e.g., $3). Expressions enclosed by parenthesis (…) are stored in the variables $1, $2 … $n.

Note that spaces within a regular expression must be enclosed by parenthesis, as shown in the first example below:

```
body.sdp regex (AVP 8)
body.sdp regex avp
```

This feature provides the following main benefits:

- The device does not need to know the SIP header name or structure.
- The sub-expressions can be used in the manipulation action. All that is required is to set the action (for example, add, modify, etc.) and then reference the sub-expression you want to use as the value.

Regex is supported in the following tables and corresponding fields:

Table 4-1: Configuration Tables and Relevant Fields

Table	Fields
Message Manipulations Table	■ 'Condition' ■ 'Action Value'
Message Conditions Table	■ 'Condition'

Table	Fields
Pre-Parsing Manipulation Table	■ 'Pattern' ■ 'Replace-With'
Call Setup Rules Table	■ 'Condition' ■ 'Action Value'

Syntax:

```
<$n>
```

Where <$n> is used to reference a resulting sub-expression after executing a regex in a condition; where n is an integer referencing the sub-expression.

- The regex pattern ".*" for multi-line match is supported.
- To concatenate, use the + operator, e.g., $1 + 'some text' + $3
- Rand function can be used in the 'Replace-With' field (as a sub-value), for example, $1 + rand.number.1.10.
- Double backslash \\ can be used to state a new line (in 'Replace-With' field), for example, to replace a pattern with multi-line text.

Regex Basic Examples

The following table provides configuration examples for using regular expressions in the Message Manipulations table.

Table 4-2: Regex Examples for Message Manipulation, Message Conditions and CSR

Message Type	Condition	Action Subject	Action Type	Action Value
invite.request	header.diversion.0 regex (<.*)(;urlparam=[a-z]*)(.*>)	header.diversion.0	**Modify**	$1+$3
invite.request	header.diversion.0 regex (<.*)(;urlparam=[a-z]*)(.*>)	header.diversion.0	**Add**	$1 + ';mynewparam=good' + $3

Message Type	Condition	Action Subject	Action Type	Action Value
invite.response.100	header.via regex (SIP/2.0/UDP)(.*);branch=(.*)	header.thebranch	**Add**	$3
subscribe	header.to regex (.*)(1001)(.*)@(.*)>	header.to	**Modify**	$1+$3+'8@'+$4

Table 4-3: Regex Examples for Pre-Parsing Manipulation Rules

Message Type	Pattern	Replace-With	Explanation
invite.request	From: *<sip:([^@]+)(@\S*)	'From: <sip:' + '1000' + $2	Replaces user part (if exists) in From header URL with 1000, for INVITE only.
invite.request	(From: *<sip:)([^@]+)(@\S*)	$1 + '1000' + $3	Same as above.
any.response	Refer-To: *(sip:\S*)	'Refer-To: <' + $1 + '>'	Encloses Refer-To header value with angled brackets <...>, for responses only.
	From: *([^"\r\n]*)(<sip:\S*)	'From: "' + $1 + '" ' + $2	Adds quotes to the From

Message Type	Pattern	Replace-With	Explanation
			header's display name (if exists).
cancel.request	(CANCEL sip:)(\d*)(@)(.*)(To:<sip:)(\d*)(@)(.*)	$1+$6+$3+$4+$5+$6+$7+$8	Replaces the user part of the Request-Uri header URL with user part of the To header URL, for SIP CANCEL messages only.

Regex Detailed Examples

Below are detailed examples of using regex for SIP message manipulation:

- Example 1 - Number range matching and manipulation:
  - Required manipulation:

    When the source number has prefix 30 to 40 and a digit (e.g., 3122), it needs to be changed to 2312. The last digit of the original phone number is removed (i.e., 2, leaving the number as 312) and the result is prefixed with 2.

    - Old header:

      ```
      To: <sip:3122@10.132.10.100;user=phone
      ```

    - New header:

      ```
      To: sip:2312@10.132.10.100;user=phone
      ```

  - Manipulation rule:

Condition	Action Subject	Action Type	Action Value
header.to regex (<.*)([3-4][0-	header.to	**Modify**	$1+'2'+$2+$3+'@'+$5

Condition	Action Subject	Action Type	Action Value
9])(.*)(\d)@(.*>)			

- Explanation:

  Dialing 3122 creates the following sub-expressions:

  - 1: <sip:
  - 2: 31
  - 3: 2
  - 4: 2
  - 5: 10.132.10.100;user=phone>

■ Example 2 - Manipulation based on source and destination number:

- Required manipulation:

  If the destination number has prefix 6, 7, or 8 (e.g., 85262146) and the source number has prefix 2001, then remove the first five digits (e.g., 85262) from the destination number and add 3 as the prefix (e.g., 3146).

  - Old header:

    From: <sip:20011234@10.132.10.100;user=phone>;tag=XINPYDPROEOREGEIHUHF
    To: sip:85262146@10.132.10.100;user=phone

  - New header:

    From: <sip:20011234@company246.com;user=phone>;tag=1c13519
    To: sip:3146@company244.com

- Manipulation rules:

Condition	Action Subject	Action Type	Action Value
header.to regex <sip:([6-8][1-9]{4})(.*)@	var.call.dst.0	**Modify**	'3'+$2

Condition	Action Subject	Action Type	Action Value
(.*>)			
header.from regex 2001	header.to.url.user	**Modify**	var.call.dst.0

- Explanation:

  These rules are slightly complex as both the To and From headers are inspected. This rule executes

  - If the dialed number is prefixed with a number 6-8 (inclusive)
  - If the calling party number is prefixed with 2001

  If these conditions exist, then:

  - Remove the first five digits of the dialled string.
  - Prefix the result with the digit 3.

  The first rule matches a dialled number that occurs in the To header (e.g., 85262146). If a match occurs, it uses a variable to store the remaining three digits and adds the digit 3 as the prefix. The second rule inspects the From header. If it contains the string 2001, then the user part of the To header is modified with the prepared variable. For example, the user (at 20011234) dials 85262146, which generates the following substring from the first rule:

  - $1 85262
  - $2 146
  - $3 10.132.10.100;user=phone>

This configuration isolates the last three digits in the dialed number and prefixes them with '3'. The variable now is set to '3146'. The second rule does not use sub-expressions. It simply searches for 2001 in the From header and if there is a match the user part of the To header is manipulated using the standard manipulation syntax.

- Example 3 - Manipulation of SDP:
  - Manipulation required:

    To change the packet period in the SDP.

  - Manipulation rule:

Condition	Action Subject	Action Type	Action Value
body.sdp regex	body.sdp	**Modify**	$1+'a=ptime:10'+$3

Condition	Action Subject	Action Type	Action Value
`(.*)(a=ptime:20)(.*)`			

- Explanation:

  This rule matches everything up to the a=ptime in the SDP body as $1, and stores as $3 everything after the 0 in the ptime attribute line. This is used as the closing \r\n in the SDP body. The modify action then refers to the sub-expressions $1 and $3, but does not make use of $2, instead replacing it with a=ptime:10.

■ Example 4 – Manipulation of SDP:

- Manipulation rule:

Condition	Action Subject	Action Type	Action Value
`body.sdp regex (.*)(m=audio)(.*)(m=audio)(.*)`	`body.sdp`	**Modify**	`$1+$2+$3`

- Explanation:

  The dollar "$" values represent each condition that is enclosed by parentheses:

  - (.*) = $1
  - (m=audio) = $2
  - (.*) = $3
  - (m=audio) = $4
  - (.*) = $5

  The 'Value' field means keep $1, $2, and $3 (and remove $4 and $5). The lines in the SDP represented by each $ is shown below:

Original SDP (color-coded to denote $ values):

```
v=0
o=mv 1980150132 244692855 IN IP6 2600:1f16:c96:aa00:951f:f946:4ebf:ef8c
s=-
c=IN IP6 2600:1f16:c96:aa00:951f:f946:4ebf:ef8c
t=0 0
a=group:ANAT 1 2
m=audio 11090 RTP/AVP 0 8 101
c=IN IP6 ::
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
```

```
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=mid:1
m=audio 11090 RTP/AVP 0 8 101
c=IN IP4 0.0.0.0
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=mid:2
```

SDP after Manipulation:

```
v=0
o=mv 1980150132 244692855 IN IP6 2600:1f16:c96:aa00:951f:f946:4ebf:ef8c
s=-
c=IN IP6 2600:1f16:c96:aa00:951f:f946:4ebf:ef8c
t=0 0
a=group:ANAT 1 2
m=audio 11090 RTP/AVP 0 8 101
c=IN IP6 ::
a=rtpmap:0 PCMU/8000
a=rtpmap:8 PCMA/8000
a=rtpmap:101 telephone-event/8000
a=fmtp:101 0-15
a=mid:1
```

Variables for Copying Data between Messages

You can use variables in SIP message manipulation rules to copy specific information (data) from one message to another. Information from one message is copied to a variable and then information from that variable is copied to any subsequent message. The device can store information in local (call) or global variables.

To store data in a variable, add the name of the variable in the 'Action Subject' field and set the 'Action Type' to **Modify**. To retrieve data from a variable, add it in the 'Action Value' field, or use it as a condition value in the 'Condition' field.

- All variable types described in this section support up to 15 variables per entity (e.g., a specific call) for Mediant Software 8 GB and higher, and up to 10 for all other devices. For each variable type per entity, all the used variables together can contain only up to 1,500 characters.
- Variable names can include alphanumeric characters and hyphens (-).
- Variable values can be any string.

Call Variables

A call variable stores specific information of a SIP message on a per call basis and changes when a new call is made (i.e., stored only throughout the lifetime of a specific call). The stored information can be, for example, the value of the SIP Contact header.

Local variables can be used per call: src (source) or dst (destination) references (see note below) which can be stored in the call leg.

Syntax:

```
var.call.src|dst.<Variable Name>
```

where:

- *src* denotes the call source variable.
- *dst* denotes the call destination variable.
- *<Variable Name>* specifies the name of the variable. You can also use hard-coded variable names for the following functionality:
  - Retrieving values from SIP headers for specific CDR fields when customizing the CDR format (see Call Variable for Obtaining CDR Field Value from SIP Header on the next page).
  - Playing a background tone (defined in the PRT file) to the call parties in an SBC call (see Call Variable for Playing Background Tone on page 74).
  - Defining maximum call duration (see Call Variable for Maximum Call Duration on page 75).

- The `var.call.src` and `var.call.dst` depend on the context in which they are used – incoming or outgoing leg. The `var.call.dst` is obtained from the current leg; `var.call.src` is obtained from the peer leg. Therefore, if `var.call.src` is used for the incoming leg (IP Group's 'Inbound Message Manipulation Set' parameter), the variable is from the outgoing leg, which is the peer of the current leg. If `var.call.src` is used for the outgoing leg (IP Group's 'Outbound Message Manipulation Set' parameter), the variable is from the incoming leg (source), which is the peer of the current leg.
- Information stored in the call variable is only valid for the duration of the call.

For example:

1. Store a value in a call variable (store the subject URI parameter value of the To header):

```
MessageManipulations 0 = 0, Invite.Request, , var.call.dst.My-Call, 2,
header.to.url.param.subject, 0;
```

2. Use the stored value (allocate a Subject header for the 200 OK response for the same call and assign it the stored value):

```
MessageManipulations 0 = 0, Invite.response.200, , header.subject, 0,
var.call.dst.My-Call, 0;
```

The following table provides additional configuration examples of using call variables in Message Manipulation rules.

Table 4-4: Examples of Call Variables

Message Type	Condition	Action Subject	Action Type	Action Value
`invite`	`param.message.sdp.rtpmode=='sendrecv'`	`Var.Call.Src.My-Call`	**Modify**	`'1'`
`invite.response.200`	`var.call.dst.My-Call=='1'`	`Param.Message.Sdp.rtpmode`	**Modify**	`'sendonly'`

Call Variable for Obtaining CDR Field Value from SIP Header

For call variables, you can use hard-coded variable names for retrieving values from SIP headers in SIP messages for specific CDR fields when customizing the CDR format:

- **User-Defined CDR Fields:** The device provides up to five user-defined CDR fields called 'Var Call User Defined <1-5>', which you can include in the generated CDR. To obtain the value for these fields from SIP messages, use the following syntax:

  ```
  var.call.src|dst.userdefined<1-5>
  ```

  Once you have configured the Message Manipulation rule, you need to customize the CDR in the SBC CDR Format table or Gateway CDR Format to include the 'Var Call User Defined <1-5>' field. Select the corresponding **Var Call User Defined <1-5>** option from the 'Field Type' drop-down list.

> For Gateway calls, only the call variable `var.call.dst.userdefined<1-5>` is applicable (because there is only one "leg" in the Gateway application). Therefore, to apply the Message Manipulation rule to the incoming or outgoing SIP message, use the [GWInboundManipulationSet] or [GWOutboundManipulationSet] ini file parameters, respectively.

- **Tenant ID CDR Field:** (SBC application only) To obtain the value for the 'Tenant ID' CDR field from the SIP message (typically used by OVOC), use the following syntax:

  ```
  var.call.src|dst.TenantId
  ```

Once you have configured the Message Manipulation rule, you need to customize the CDR in the SBC CDR Format table to include the 'Tenant ID' CDR field. Select **Tenant ID** from the 'Field Type' drop-down list.

The call variable `var.call.src|dst.TenantId` is applicable only to the SBC application.

The following examples use the URL of the Contact header in the SIP INVITE message as the value for the 'Tenant ID' and 'User-Defined1' CDR fields:

Table 4-5: Examples of Using Call Variables for Obtaining CDR Field Values from SIP Headers

Message Type	Condition	Action Subject	Action Type	Action Value
invite	-	Var.Call.Src.TenantId	**Modify**	Header.Contact.URL
invite	-	Var.Call.Dst.UserDefined1	**Modify**	Header.Contact.URL

Call Variable for Playing Background Tone

You can configure the device to play a background tone (defined in the PRT file) to the call parties in an SBC call. The tone can be played to one or both of the call parties (caller and/or callee). When played to both parties, the tone is played simultaneously. The device can play the tone after call establishment or during early media. This feature can be useful, for example, to indicate that a call is being recorded.

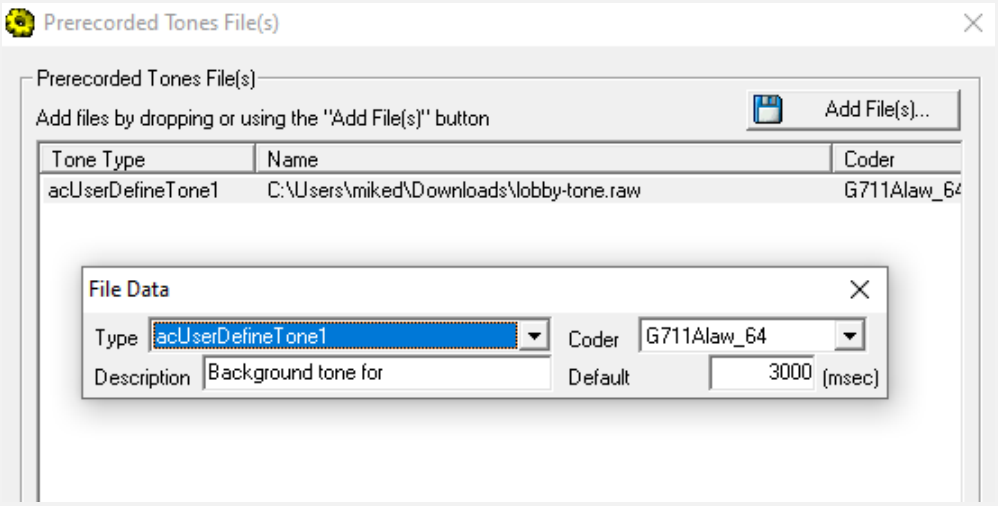
For playing background tones, you need to configure a Message Manipulation rule with the following special options:

- 'Action Subject' field: `Var.Call.Dst|Src.PlayBackgroundTone` - enables background tone on the configured side (Dst) or peer side (Src)
- 'Action Value' field: `<side>,<tone ID>,<time between play>`, where:
  - *<side>* defines the call party to which the device plays the tone – **both** (caller and callee) or **single** (only on side according to `Var.Call.Dst|Src.PlayBackgroundTone`).
  - *<tone ID>* defines the user-defined tone to play, by index in the PRT file (acUserDefineTone<ID>).
  - *<time between play>* defines the duration (1 to 600,000 msec) of no tone play between each play of tone. For example, you can use this option to play a beep every 3 seconds (periodically). Alternatively, if you want the device to play the tone continuously, don't configure this option.

  For example:

- The value "both,5,3000" means that the device plays the acUserDefineTone5 tone (in the PRT file) every 3000 msec to both caller and callee.
- The value "single,5" means that the device plays the acUserDefineTone5 tone (in the PRT file) continuously to the configured party only.

- If you want the device to play the tone only once, make sure that when you're converting the recorded tone to a file (PRT) that's loadable to the device using the DConvert utility, that you configure the 'Default' field to a non-zero value:



- For devices in HA, the device resumes playing the tone after a switchover.
- This feature is applicable only to the SBC application.

Call Variable for Maximum Call Duration

You can use special hard-coded call variables in Message Manipulation rules to configure the maximum call duration. This call duration is applied to calls belonging to the IP Group to which the Message Manipulation rule (Set ID) is associated.

- The value of the call variable overrides the call duration value configured by the following configuration parameters:
  - ✓ Global parameter [SBCMaxCallDuration].
  - ✓ IP Profile parameter 'SBC Max Call Duration'.
- This call variable is applicable only to the SBC application.

Maximum call duration in Message Manipulation rules is done as follows:

■ Configure the 'Action Subject' field to one of the following call variables (depending on leg):

```
Var.Call.Dst.MaxDuration
```

```
Var.Call.Src.MaxDuration
```

For an explanation on the difference between `Var.Call.Dst` and `Var.Call.Dst`, see Call Variables on page 72.

- Configure the 'Action Type' field to **Modify**.
- Configure the 'Action Value' field to the maximum call duration (in minutes). The value must be enclosed with single quotes (e.g., '30'). A value of '0' means unlimited duration.

To apply the maximum call duration to both incoming and outgoing calls for the IP Group, assign the Message Manipulation rule (Set ID) to both the 'Inbound Message Manipulation Set' and 'Outbound Message Manipulation Set' fields in the IP Groups table. Alternatively, you can assign the Message Manipulation Set ID to the 'Outbound Message Manipulation Set' field only, where the Manipulation Set ID can include one or two Message Manipulation rules:

- Two rules - one for the INVITE request (`Invite.Request`) and another for the INVITE response (`Invite.Response`).
- One rule - with the 'Message Type' field configured to `any` (applies only to INVITE).

The following example configures a Message Manipulation Set ID with two Message Manipulation rules that define maximum call duration as 10 minutes for both incoming and outgoing calls:

Table 4-6: Configuration Example of Call Variable for Maximum Call Duration

Message Type	Condition	Action Subject	Action Type	Action Value
`Invite.Request`	-	`Var.Call.Dst.MaxDuration`	**Modify**	`'10'`
`Invite.Response`	-	`Var.Call.Dst.MaxDuration`	**Modify**	`'10'`

Global Variable

Global variables are similar to call variables, but they do not change as new calls are made (i.e., their lifetime is not restricted to the duration of a call).

Syntax:

```
var.global.<Variable Name>
```

where, <Variable Name> specifies the name of the global variable.

For example:

- Store a value in a global variable: Stores the Priority header of the INVITE with 'company' in the host part of the From header:

```
MessageManipulations 0 = 0, Invite.Request, header.from.url.host == 'company', var.global.My-Global, 2, header.priority, 0;
```

- Use the stored value: Assigns the same priority as the INVITE request to SUBSCRIBE requests arriving with 'company' in the host part of the From header:

```
MessageManipulations 0 = 0, Subscribe.request, header.from.url.host == 'company', header.priority, 0, var.global.My-Global, 0;
```

The following table provides additional configuration examples of using variables in Message Manipulation rules.

Table 4-7: Example of Global Variables

Message Type	Action Subject	Action Type	Action Value
invite	var.global.My-Global	**Modify**	'Custom UA'

Session Variable

Session variables can be preserved in any ongoing leg in the session, for example, in an call session with forking calls, in a call which had a locally handled blind transfer, etc. The value of the variable remains the same in all existing legs and in new legs of the session context.

Syntax:

```
var.session.<Variable Name>
```

where, <Variable Name> specifies the name of the variable.

For example (using SIPRec):

For an IP-to-Tel call, the INVITE message of the recorded IP call contains the header, X-credit-card (e.g., X-credit-card: 123456789). When the device sends an INVITE to the SIPRec server (SRS), it is required to include the content (value) of this header (e.g., 123456789). To do this, you need to configure two Message Manipulation rules:

1. For the recorded call: This rule stores the content of the X-credit-card header in the variable, var.session.SIPRec.
2. For the SRS leg: This rule adds a new header, X-credit-card with the contents of the variable (var.session.SIPRec) to the INVITE sent to the SRS.

Table 4-8: Example of Session Variables

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite.request	header.X-credit-card exists	var.session.SIPRec	**Modify**	header.X-credit-card	For the recorded call
invite.request	var.session.SIPRec != ''	header.X-credit-card	**Add**	var.session.SIPRec	For the SRS leg

Registered User Variable

The user variable stores information for each far-end user registered with the device and maintains it as long as the user is registered. The user variables can be accessed from all SIP dialogs which the user is involved in.

Syntax:

```
var.user|peer-user.<Variable Name>
```

where:

- user denotes the user of the current leg
- peer-user denotes the user of the peer leg
- <Variable Name> specifies the name of the variable

Usages:

- Inbound and outbound message manipulation
- Call Setup Rules (only var.user)
- IP-to-IP Routing table:
  - 'Message Condition' field (only var.user and read-only)
  - 'Internal Action' field (only var.user and read-only)

- This section is applicable only to the SBC application.
- The user variable cannot be used in Outbound Manipulation for operations that are terminated before Classification, Manipulation and Routing (responses for REGISTER termination, Reply internal action, CMR failure).
- When a registered user expires and the device generates and sends an un-REGISTER to the proxy, you can configure outbound manipulation for the Server-type IP Group (i.e., proxy) using the param.ipg.src|dst.<x> syntax. For this stage, you can also use the var.peer-user variable.

The following table provides a configuration example of user variables in Message Manipulation rules.

- Index 1: When the REGISTER response (200 OK) contains the Proprietary-Header header, the device stores it in a user variable ("myheader").
- Index 2: This variable ("myheader") is later used to add a Proprietary-Header header to subsequent 200 OK responses for REGISTER messages (that do not contain the header) sent to that user, without requiring it to be re-sent by the registrar.

Table 4-9: Examples of User Variables

Index	Message Type	Condition	Action Subject	Action Type	Action Value
1	register.response.200	header.proprietary-header exists	var.user.myheader	**Modify**	header.proprietary-header
2	register.response.200	header.proprietary-header !exists	header.proprietary-header	**Add**	var.user.myheader

Specifying Tone to Play Upon Call Connect

Call variables (described in Section Call Variables on page 72) can be used to specify the tone to play upon call connection (after SIP 200 OK). The tone is defined in the loaded Prerecorded Tone (PRT) file. When the tone finishes playing, the call is connected and the call parties can begin talking.

This is done using the following variable:

```
var.call.src|dst.PlayToneOnConnect
```

The following table provides a configuration example of using the play-tone call variable in Message Manipulation rules. The rule instructs the device to play the tone at Index #105 in the PRT file, to the destination call party.

Table 4-10: Example of Call Variable for Specifying Tone to Play

Message Type	Condition	Action Subject	Action Type	Action Value
`invite.request`	`Header.From contains '100'`	`var.call.dst.PlayToneOnConnect`	**Add**	`'105'`

For more information on this feature, refer to the device's User's Manual.

Functions

You can use pre-defined functions in message manipulations for special operations such as changing a returned value from lower case to upper case (see example below).

Functions can be used in fields where manipulation terms are read-only:

- Message Manipulations: 'Condition' and 'Action Value' fields
- Pre-Parsing Message Manipulation Set: "Replace-With" manipulation term
- Call Setup Rules: 'Request Key', 'Condition', and 'Action Value' fields

Syntax:

```
Func.<FunctionName>(<Message Manipulation Term>)
```

The following table describes the supported functions.

Table 4-11: Function Descriptions

Function	Description
Encrypt	Encrypts the value of the specified SIP header, using AES-256 encryption key algorithm. The key is configured using the [EncryptKeyAES256] parameter. **Note:** This function is intended for SIP headers that are not used by the device for classification or routing. For example, you may want to encrypt the value of a proprietary SIP header called "P-Access-Network-Info" that may contain sensitive information.
Decrypt	Decrypts the AES-256 encrypted value of the specified SIP

Function	Description
	header. The key is configured using the [EncryptKeyAES256] parameter. **Note:** This function is intended for SIP headers that are not used by the device for classification or routing. For example, you may want to encrypt the value of a proprietary SIP header called "P-Access-Network-Info" that may contain sensitive information.
Decrement	If the message manipulation term is evaluated to be integer x, the function returns x – 1. If the term is evaluated to be a non-integer, no action is done.
Increment	If a message manipulation term is evaluated to be integer x, the function returns x + 1. If the term is evaluated to be a non-integer, no action is done.
Length	Returns the length of the evaluated message manipulation term string.
To-Lower	Changes characters in the evaluated message manipulation term to lowercase.
To-Upper	Changes characters in the evaluated message manipulation term to uppercase.
URL-Decode	Decodes evaluated Message Manipulation term characters according to RFC 3986 Section 2. Translates each encoded character in the string to the character itself. Example: 'User%40audiocodes.com' is decoded to 'User@audiocodes.com'.
URL-Encode	Encodes evaluated message manipulation term characters according to RFC 3986 Section 2. It replaces reserved characters in a string with their hexadecimal representation (preceded by "%"). All characters are encoded except the following (considered unreserved): ■ Alphabet characters (A-Z, a-z) ■ Digit characters (0-9) ■ Hyphen "-" ■ Underscore "_" ■ Dot "."

Function	Description
	■ Tilde "~" For example, "user@abc.com" is encoded to "user%40abc.com".
UUID-Generate	Generates a random Universally Unique Identifier (UUID) value.

Currently, concatenated message manipulation terms inside the function's parentheses is not supported. For example, the following is not supported: Func.To-Upper (header.form.url.user + '@' + header.to.url.host). However, for fields supporting concatenation, you can concatenate the function as shown in the following example:
Func.To-Upper(header.form.url.user) + '@' + Func.To-Upper(header.to.url.host)

Examples:

■ The following Message Manipulation rule encrypts the value of the header "My-Identifier" in the outgoing SIP INVITE message:

Message Type	Condition	Action Subject	Action Type	Action Value
`invite.request`	-	`Header.My-Identifier`	**Modify**	`Func.Encrypt (Header.My-Identifier)`

■ The following Message Manipulation rule adds the header "My-Host" to the outgoing SIP message, whose value is set to the source host, which is converted into upper case letters, using the function To-Upper:

Message Type	Condition	Action Subject	Action Type	Action Value
`invite.request`	-	`Header.My-Host`	**Add**	`Func.To-Upper (Param.Call.Src.Host)`

If the above rule is used and the host part in the From header of the SIP message is "JohnB":

```
From: <SIP:1000@JohnB>; tag-1c1000228485
```

After manipulation, the following header with the host value in upper case ("JOHNB") is added to the outgoing message:

From: <SIP:1000@JohnB>; tag-1c1000228485
My-Host: JOHNB

- The following Call Setup rule performs an ENUM query on an ENUM server for the source user and if found, it returns a string from the URL that is defined by regex (the string after "us-ascii,"), and then converts encoded characters in the string and adds it as the name in the From header.

Request Type	Request Key	Condition	Action Subject	Action Type	Action Value
Enum	Param.Call.Src.User	Enum.Found Exists AND Enum.Result.Url regex 'us-ascii,(.*)'	Header.From.Name	**Modify**	Func.URL-Decode($1)

If the above rule is used and the returned URL from the ENUM query is:

: pstndata:cnam/7039532959;;charset=us-ascii,John%20Bow

The rule then extracts and decodes " John%20Bow" to "John Bow" and adds it to the From header:

From: "John Bow" <sip:+61424795803@abc.rob.com.au>;tag=1c1474248679

- The following Message Manipulation rule adds the header "My-Identifier" to the outgoing SIP message, whose value is a randomly generated UUID, using the function UUID-Generate:

Message Type	Condition	Action Subject	Action Type	Action Value
invite.request	-	Header.My-Identifier	**Add**	Func.UUID-Generate

ISUP Body Manipulation

For certain ISUP call actions, see also Special Actions using X-AC-Action SIP Header on page 102.

SIP Method	ISUP Message Type	Parameter	Field	Syntax
INVITE	IAM	Called party number	Number Plan (see Number Plan on page 176)	`body.isup.iam.called_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.iam.called_num.type`
			Address signal (string of up to 50 characters)	`body.isup.iam.called_num.digits`
			Internal Network Number indicator (INN) ■ 0: routing to internal number allowed ■ 1: (default) routing to internal number not allowed	`body.isup.iam.called_num.inn`
		Calling party number	Number Plan (see Number Plan on page 176)	`body.isup.iam.calling_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.iam.calling_num.type`
			Address	`body.isup.iam.calling_`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			presentation restricted indicator (see Address Presentation Restricted Indicator on page 183)	`num.presentation`
			Screening indicator (see Screen on page 182)	`body.isup.iam.calling_num.screening`
			Address signal (string of up to 50 characters)	`body.isup.iam.calling_num.digits`
		Original Called Number	Number Plan (see Number Plan on page 176)	`body.isup.iam. original_called_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.iam. original_called_num.type`
			Address presentation restricted indicator (see Address Presentation Restricted Indicator on page 183)	`body.isup.iam.original_called_num.presentation`
			Address signal (string of up to 50 characters)	`body.isup.iam.original_called_num.digits`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
		Generic Number	Number qualifier indicator (see Q.763.3.26)	`body.isup.iam.generic_num.qualifier`
			Number Plan (see Number Plan on page 176)	`body.isup.iam.generic_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.iam.generic_num.type`
			Address presentation restricted indicator (see Address Presentation Restricted Indicator on page 183)	`body.isup.iam.generic_num.presentation`
			Screening indicator (see Screen on page 182)	`body.isup.iam.generic_num.screening`
			Address signal (string of up to 50 characters)	`body.isup.iam.generic_num.digits`
		Location Number	Number Plan (see Number Plan on page 176)	`body.isup.iam.location_num.plan`
			Number Type (see Number	`body.isup.iam.location_num.type`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			Type on page 177)	
			Address presentation restricted indicator (see Address Presentation Restricted Indicator on page 183)	`body.isup.iam.location_num.presentation`
			Screening indicator (see Screen on page 182)	`body.isup.iam.location_num.screening`
			Address signal (string of up to 50 characters)	`body.isup.iam.location_num.digits`
			Internal Network Number indicator (INN) ■ 0: routing to internal number allowed ■ 1: (default) routing to internal number not allowed	`body.isup.iam.location_num.inn`
		Redirecting number	Number Plan (see Number Plan on	`body.isup.iam.redirecting_num.plan`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			page 176)	
			Number Type (see Number Type on page 177)	`body.isup.iam.redirecting_num.type`
			Address presentation restricted (see Address Presentation Restricted Indicator on page 183)	`body.isup.iam.redirecting_num.presentation`
			Address signal (string of up to 50 characters)	`body.isup.iam.redirecting_num.digits`
		Redirection information	Redirecting reason (see Reason (Diversion) on page 177)	`body.isup.iam.redirect.info.reason`
			Original Redirect reason - values 1, 2, and 3 (see Redirect Reason on page 188)	`body.isup.iam.redirect_info.orig_reason`
			Redirection Counter Number of redirections the call has undergone expressed as a	`body.isup.iam.redirect_info.counter`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			number from 1 to 5.	
			Redirecting Indicator (see Q.763.3.45)	`body.isup.iam.redirect_info.indicator`
		Forward call indicator (see Q.763 3.23)	National/international call indicator	`body.isup.iam.fci.InterationalInd`
			End-to-end method indicator	`body.isup.iam.fci.End2EndMethod`
			Interworking indicator	`body.isup.iam.fci.Interworking`
			End-to-end information indicator	`body.isup.iam.fci.End2EndInformation`
			ISDN user part indicator	`body.isup.iam.fci.IsdnUserPartIndicator`
			ISDN user part preference indicator	`body.isup.iam.fci.IsdnUserPartPreference`
			ISDN access indicator	`body.isup.iam.fci.IsdnAccess`
			SCCP method indicator	`body.isup.iam.fci.SCCP`
		Transmission medium requirement (see Transmission Medium Requireme		`body.isup.iam.tmr`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
		nt on page 183)		
		Calling party's category (see Called Party Category Indicator on page 184)		body.isup.iam.cpc
		Hop Counter (1 to 31)		body.isup.iam.hop_counter
		Access transport Low layer compatibility information element (see 4.5.19 of Recommendation Q.931)	Information transfer rate	body.isup.iam.access_transport.transfer_rate
			User information layer 1 protocol	body.isup.iam.access_transport. layer1_protocol
			User rate	body.isup.iam.access_transport.user_rate
		User service information (see 3.57 of Q.763)	Information transfer capability	body.isup.iam.user_service. transfer_capability
			Information transfer rate	body.isup.iam.user_service.transfer_rate
			User information Layer 1 protocol (usually used	body.isup.iam.user_service.user_info_l1

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			for analogue modem calls)	
			Layer ident (usually used for analogue modem calls)	`body.isup.iam.user_service.layer_id`
First 18x	ACM	Backward call indicator	Charge indicator (see Charge Indicator on page 183)	`body.isup.acm.bci.charge`
			Called party's status indicator (see Called Party Status Indicator on page 184)	`body.isup.acm.bci.status`
			Called party's category indicator (see Called Party Category Indicator on page 184)	`body.isup.acm.bci.cpc`
			End-to-end method indicator (see Q.763.3.5)	`body.isup.acm.bci.End2EndMethod`
			Interworking indicator (see Q.763.3.5)	`body.isup.acm.bci.Interworking`
			End-to-end information	`body.isup.acm.bci.End2EndInformation`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			indicator (see Q.763.3.5)	
			ISDN user part indicator (see Q.763.3.5)	`body.isup.acm.bci.IsdnUserPartIndicator`
			Holding indicator (see Q.763.3.5)	`body.isup.acm.bci.HoldingIndicator`
			ISDN access indicator (see Q.763.3.5)	`body.isup.acm.bci.IsdnAccess`
			Echo control device indicator (see Q.763.3.5)	`body.isup.acm.bci.Echo`
			SCCP method indicator (see Q.763.3.5)	`body.isup.acm.bci.SCCP`
		Optional Backward Call indicators	In-band information indicator (see Q.763.3.37)	`Body.isup.acm.obci.Inband`
			Call diversion may occur indicator (see Q.763.3.37)	`body.isup.acm.obci.diversion`
			Simple segmentation indicator (see Q.763.3.37)	`body.isup.acm.obci.segmentation`
			MLPP user	`body.isup.acm.obci.MLPP`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			indicator (see Q.763.3.37)	
Not First 18x	CPG	Event Information, can be used only in condition, (see Event Information on page 184)		`body.isup.cpg.event_info`
		Backward Call Indicator (to send it cpc, must be set manually by message manipulation)	Charge indicator (see Charge Indicator on page 183)	`body.isup.cpg.bci.charge`
			Called party's status indicator (see Called Party Status Indicator on page 184)	`body.isup.cpg.bci.status`
			Called party's category indicator (see Called Party Category Indicator on page 184)	`body.isup.cpg.bci.cpc`
			End-to-end method indicator (see Q.763.3.5)	`body.isup.cpg.bci.End2EndMethod`
			Interworking	`body.isup.cpg.bci.Interwor`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			indicator (see Q.763.3.5)	`king`
			End-to-end information indicator (see Q.763.3.5)	`body.isup.cpg.bci.End2EndInformation`
			ISDN user part indicator (see Q.763.3.5)	`body.isup.cpg.bci.IsdnUserPartIndicator`
			Holding indicator (see Q.763.3.5)	`body.isup.cpg.bci.HoldingIndicator`
			ISDN access indicator (see Q.763.3.5)	`body.isup.cpg.bci.IsdnAccess`
			Echo control device indicator (see Q.763.3.5)	`body.isup.cpg.bci.Echo`
			SCCP method indicator (see Q.763.3.5)	`body.isup.cpg.bci.SCCP`
		Optional Backward Call Indicators (sent only if at least one of the fields is explicitly set by a message manipulation rule)	In-band information indicator (see Q.763.3.37)	`body.isup.cpg.obci.inband`
			Call diversion may occur indicator (see Q.763.3.37)	`body.isup.cpg.obci.diversion`
			Simple segmentation indicator (see	`body.isup.cpg.obci.segmentation`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			Q.763.3.37)	
			MLPP user indicator (see Q.763.3.37)	`body.isup.cpg.obci.mlpp`
200 OK on INVITE	ANM	Connected number	Number Plan (see Number Plan on page 176)	`body.isup.iam.connected_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.iam.connected_num.type`
			Address presentation restricted (see Address Presentation Restricted Indicator on page 183)	`body.isup.iam.connected_num.presentation`
			Address signal (string of up to 50 characters)	`body.isup.iam.connected_num.digits`
200 OK on INVITE	CON (for Spiro u variant only, sent if SIP 18x wasn't sent	Backward call indicator	Charge indicator (see Charge Indicator on page 183)	`body.isup.con.bci.charge`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
	before 200 reply)			

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			Called party's status indicator (see Called Party Status Indicator on page 184)	`body.isup.con.bci.status`
			Called party's category indicator (see Called Party Category Indicator on page 184)	`body.isup.con.bci.cpc`
			End-to-end method indicator (see Q.763.3.5)	`body.isup.con.bci.end2endmethod`
			Interworking indicator (see Q.763.3.5)	`body.isup.con.bci.interworking`
			End-to-end information indicator (see Q.763.3.5)	`body.isup.con.bci.end2endinformation`
			ISDN user part indicator (see Q.763.3.5)	`body.isup.con.bci.isdnuserpartindicator`
			Holding indicator (see Q.763.3.5)	`body.isup.con.bci.holdingindicator`
			ISDN access indicator (see	`body.isup.con.bci.isdnaccess`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
			Q.763.3.5)	
			Echo control device indicator (see Q.763.3.5)	`body.isup.con.bci.echo`
			SCCP method indicator (see Q.763.3.5)	`body.isup.con.bci.sccp`
INFO	FAC	Transfer number	Number Plan (see Number Plan on page 176)	`body.isup.fac.connected_num.plan`
			Number Type (see Number Type on page 177)	`body.isup.fac.connected_num.type`
			Address presentation restricted (see Address Presentation Restricted Indicator on page 183)	`body.isup.fac.connected_num.presentation`
			Address signal (string of up to 50 characters)	`body.isup.fac.connected_num.digits`
BYE, 4xx	REL	Cause value (see Cause Value on page 185)		`body.isup.rel.cause`
		Cause location		`body.isup.rel.location`

SIP Method	ISUP Message Type	Parameter	Field	Syntax
		(see Cause Location on page 188)		

Attaching ISUP Body

The syntax of message manipulation for attaching ISUP body to a SIP message is as follows:

- 'Action Subject': body.isup.xxx

  Where xxx can be one of the following:

  - IAM
  - ACM
  - CPG
  - ANM
  - SUS
  - RES
  - REL
  - RLC
  - FAC

- 'Action Type': **Add**

Below is an example of a message manipulate rule that adds the ISUP Release message to the body of SIP CANCEL request:

```
MessageManipulations 8 = "Cancel add ISUP", 1, "cancel.request", "body.isup.rel
!exists", "body.isup.rel", 0, "", 0;
```

Removing Elements from ISUP Body

All optional "number" elements (connected number, transferred number, etc.) can be removed by setting the address signal to an empty string (see the example in ISUP Message Manipulation Rules on the next page).

ISUP Optional Parameters

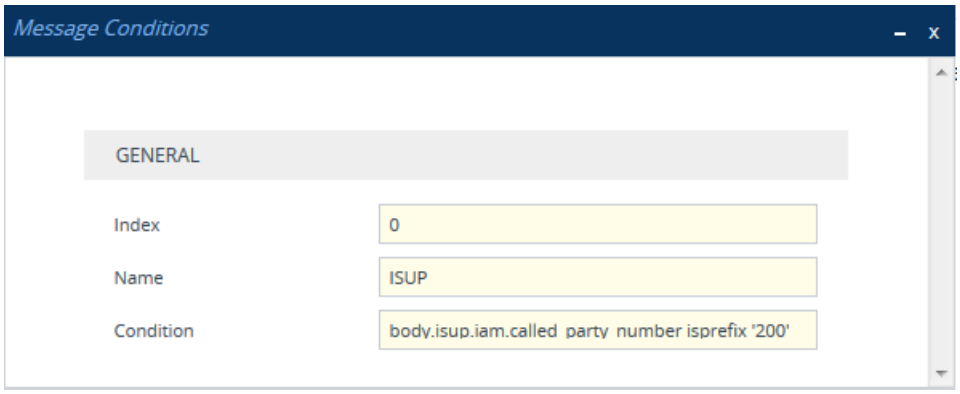
Below is th elist of optional parameters

ISUP Examples

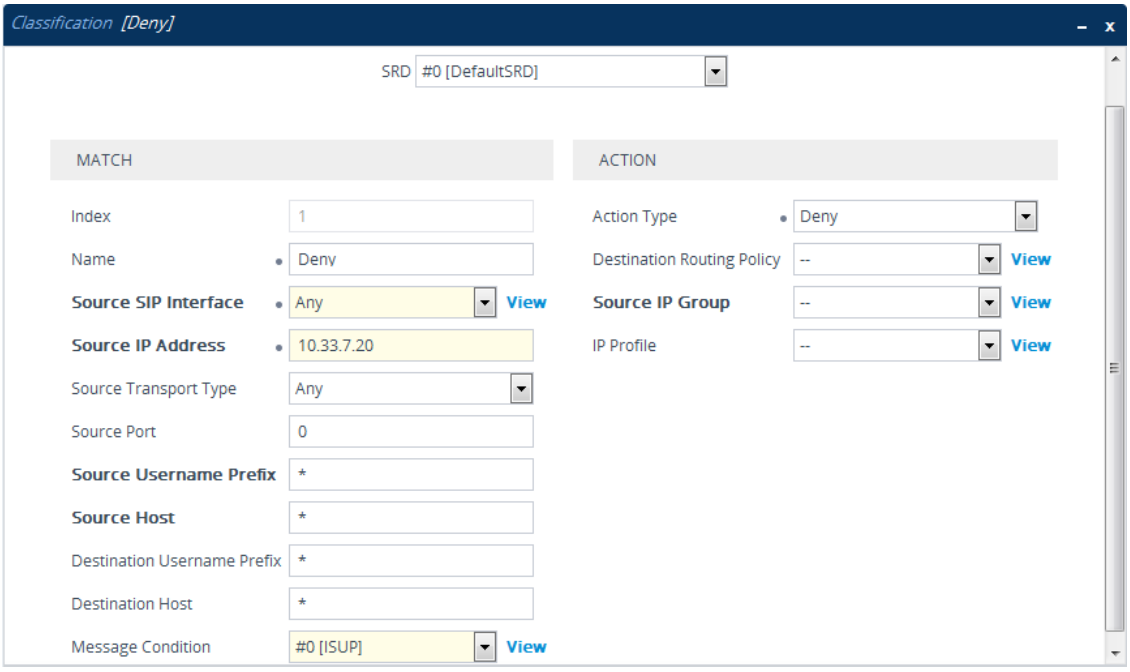
ISUP Deny Message Condition Rule

The example describes how to deny INVITE messages received from IP address 10.33.7.20 if the message contains ISUP data whose Initial Address Message (IAM) section includes a Called Party Number that begins with "200".

1. Configure a Message Condition rule in the Message Conditions table with the condition, body.isup.iam.called_party_number isprefix '200':



2. Assign the Message Condition rule to the Classification rule associated with the source of the INVITE:



ISUP Message Manipulation Rules

The example manipulates the SIP message if the incoming INVITE message includes ISUP data that contains an IAM with Calling Party Number whose Presentation is set to restricted and:

- If P-Asserted-Identity header is absent: Add P-Asserted-Identity header with value "tel:+<IAM Calling Party Number>"
- If From header is absent: Add the From header with value to "Anonymous" <sip:anonymous@anonymous.invalid>;tag=9802748
- If Privacy header is absent: Add Privacy header with value "id"

```
INVITE sip:+14085551212@gw.pstn.net SIP/2.0
  Via: SIP/2.0/TCP useragent.audiocode.com;branch=z9hG4bK-124
 To: <sip:+14085551212@audiocodes.com>
 From: "Anonymous" <sip:anonymous@anonymous.invalid>;tag=9802748
 Call-ID: 245780247857024504
 CSeq: 2 INVITE
 Max-Forwards: 68
 P-Asserted-Identity: tel:+14085264000
 Privacy: id
```

Table 4-12: ISUP Body Manipulation Rules Examples

Message Type	Condition	Action Subject	Action Type	Action Value	Row Rule
invite	body.isup exists AND body.isup.iam.calling_party_number.presentation == 'restricted' AND header.p-asserted-identity !exists	header.p-asserted-identity	Add	'tel:+' + body.isup.iam.calling_party_number.digits	Use Current Condition
invite	body.isup exists AND body.isup.iam.calling_party_number.pre	header.from	Add	"'"Anonymous" <sip:anonymous@anonymous.invalid>;tag=9802748'"	Use Current Condition

Message Type	Condition	Action Subject	Action Type	Action Value	Row Rule
	sentation == 'restricted' AND header.from !exists				
invite	body.isup exists AND body.isup.iam.calling_party_number.presentation == 'restricted' AND header.Privacy !exists	header.privacy	Add	'id'	Use Current Condition

Special Actions using X-AC-Action SIP Header

You can use AudioCodes proprietary SIP header, X-AC-Action in message manipulation rules to trigger certain actions. These actions can be used to support, for example, interworking of SIP-I and SIP endpoints for the ISUP SPIROU variant.

The following actions are supported by the X-AC-Action header:

- To disconnect a call (optionally, after a user-defined time):

```
X-AC-Action: 'disconnect'
X-AC-Action: 'disconnect;delay=<time in ms>'
```

- To resume a previously suspended call:

```
X-AC-Action: 'abort-disconnect'
```

- To automatically reply to a message without forwarding the response to the other side:

```
X-AC-Action: 'reply'
```

- To automatically reply to a message with a specific SIP response without forwarding the response to the other side:

```
X-AC-Action: 'reply;response=<response code, e.g., 200>]'
```

- To override the device's handling of SIP REFER messages for SBC calls, configured by the 'Remote Refer Mode' (IpProfile_SBCRemoteReferBehavior) parameter. The X-AC-Action header can be added to the incoming SIP REFER request using Message Manipulation rules. This is useful if you don't want the settings of this parameter to apply to all calls that are associated with the IP Profile. For example, if you configure the 'Remote Refer Mode' parameter to Handle Locally, all incoming SIP REFER requests associated with the IP Profile are terminated at the device. However, you can configure a Message Manipulation rule with the proprietary header to override this parameter setting and allow the device to forward the REFER requests as is for calls with a specific URI, for example. You can configure Message Manipulation rules to add this X-AC-Action header for REFER handling, with one of the following values:
  - To allow the device to forward the REFER as is, regardless of the 'Remote Refer Mode' parameter settings:

    ```
    X-AC-Action: 'use-config;refer-behavior=regular'
    ```

  - To allow the device to handle (terminate) the REFER request regardless of the 'Remote Refer Mode' parameter settings:

    ```
    X-AC-Action: 'use-config;refer-behavior= handle-locally'
    ```

- To switch to a different IP Profile for the call (re-INVITE only), as defined in the IP Group:

```
X-AC-Action: 'switch-profile;profile-name=<IP Profile Name>'
X-AC-Action: 'switch-profile;profile-name=<IP Profile
Name>;reason=<PoorInVoiceQuality or PoorInVoiceQualityFailure>]'
```

If the IP Profile name contains one or more spaces (e.g., "ITSP NET"), enclose the name in double quotation marks, for example:

```
X-AC-Action: 'switch-profile;profile-name="ITSP NET"'
```

The X-AC-Action header for triggering the actions *disconnect*, *abort-disconnect*, and *reply* is supported only for the following SIP messages (methods):

- re-INVITE
- UPDATE
- INFO
- REFER

The following table provides examples of the X-AC-Action header.

Table 4-13: X-AC-Action Header Manipulation Rule Example

Message Type	Condition	Action Subject	Action Type	Action Value	Description
`info.request`	`body.isup.sus exists`	`header.x-ac-action`	**Modify**	`'disconnect;delay=3000,reply'`	Disconnects a call after 3 seconds if the received SIP INFO message contains the ISUP SUS field.
`reinvite.request`	`body.sdp regex (.*)(m=audio 7550 RTP/AVP)(.*)`	`header.x-ac-action`	**Add**	`'switch-profile;profile-name=ITSP-Profile-2'`	If a re-INVITE message is received and whose media port value is 7550, the device adds the SIP

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					header "X-AC-Action: switch-profile;profile-name=ITSP-Profile-2"to the incoming re-INVITE message. As a result of receiving this manipulated message, the device starts using IP Profile "ITSP-Profile-2" instead of "ITSP-Profile-1", for the IP Group to which the rule is applied.

SIP Message Normalization

The device supports a built-in SIP message normalization feature that can be enabled per manipulation rule. This is enabled by configuring the 'Action Type' field to **Normalize**. Normalization removes unknown or non-standard SIP message elements before forwarding the message. These elements can include SIP headers, SIP header parameters, and SDP body fields. Message normalization is typically configured per SIP header, but can also be configured for all headers (including SDP).

The device normalizes the following SIP elements:

- **URL:** You can normalize the URL of SIP headers, which is enclosed in angled <...> brackets.

  For example:

  - Before normalization - URL contains a non-standard user part "phone-context=1" and an unknown parameter "UnknownUrlParam":

    ```
    <sip:+1-800-229-229;phone-context=1@10.33.2.17;user=phone;UnknownUrlParam>
    ```

  - After normalization - non-standard user part "phone-context=1" and an unknown parameter "UnknownUrlParam" are removed:

    ```
    <sip:+1800229229@10.33.2.17;user=phone>
    ```

  Configuration: To normalize **only** the URL of a SIP header (e.g., To):

  - 'Action Subject' field: **Header.To.Url**
  - 'Action Type' field: **Normalize**

- **Non-URL part:** For SIP headers that contain a URL, you can normalize everything outside of the URL (i.e., not enclosed in angled <...> brackets).

  For example:

  - Before normalization - an unknown parameter "UnknownHeaderlParam" outside the URL:

    ```
    To: <sip:100;phone-context=1@10.33.2.17;user=phone;UnknownUrlParam>;UnknownHeaderlParam
    ```

  - After normalization - unknown parameter removed:

    ```
    To: <sip:100;phone-context=1@10.33.2.17;user=phone;UnknownUrlParam>
    ```

  Configuration:

- 'Action Subject' field: **header.<header name>** (e.g., **header.to**)
- 'Action Type' field: **Normalize**

Headers:

- Alert-Info: unknown header parameters are removed
- P-Called-Party-ID: unknown header parameters are removed, URL is normalized
- P-Charging-Vector: unknown header parameters are removed
- P-Associated-URI: unknown header parameters are removed, URL is normalized
- P-Preferred-Identity: URL is normalized
- Diversion: unknown header parameters are removed, URL is normalized
- P-Asserted-Identity: URL is normalized
- Privacy: unknown header parameters are removed
- Remote-Party-ID: unknown header parameters are removed, URL is normalized
- Reason: unknown header parameters are removed
- Max-Forwards: value is changed to 70
- History-Info: unknown header parameters are removed, URL is normalized
- From: unknown header parameters are removed, URL is normalized
- To: unknown header parameters are removed, URL is normalized
- Via: unknown header parameters are removed
- Refer-To: unknown header parameters are removed, URL is normalized
- Referred-By: unknown header parameters are removed, URL is normalized
- Event: unknown header parameters are removed
- Session-Expires: unknown header parameters are removed
- Min-SE: unknown header parameters are removed
- Min-Expires: unknown header parameters are removed
- Request-URI: URL is normalized
- Contact: unknown header parameters are removed
- Subscription-State: unknown header parameters are removed

Configuration:

- 'Action Subject' field: **header.<header name>** (e.g., **header.min-expires**)
- 'Action Type' field: **Normalize**

For SIP headers that contain a URL (e.g., Contact, Diversion, and Referred-By), to normalize the URL (i.e., everything enclosed in angled <...> brackets), use the following syntax for the 'Action Subject' field: *Header.<header name>.Url* (e.g., **Header.Contact.Url**). To normalize everything except the URL (i.e., everything outside of <...>), use this syntax: *Header.<header name>* (e.g., **Header.Contact**). For more information, see the descriptions above for URL and non-URL normalization.

- **SDP Body:** Removes unnecessary SDP fields (except v=, o=, s=, c=, t=, and r=) and unknown media with all its attributes. For example, the bold is removed before sending the message:

```
v=0
o=SMG 791285 795617 IN IP4 10.33.2.17
s=Phone-Call
i=A Seminar on the session description protocol
u=http://www.example.com/seminars/sdp.pdf
e=j.doe@example.com (Jane Doe)
c=IN IP4 10.33.2.26
t=0 0
m=unknown 6000 RTP/AVP 8
a=unknown
a=sendrecv
a=ptime:20
m=audio 6000 RTP/AVP 8
a=rtpmap:8 pcma/8000
a=sendrecv
a=unknown
a=ptime:20
```

Configuration:

- 'Action Subject' field: **body.sdp**
- 'Action Type' field: **Normalize**

- **Message:** Normalization of the entire message. Headers and bodies not listed below are removed while those listed are retained and normalized (if necessary and if listed as supported for normalization, as previously mentioned):
  - Headers:
    - Request-URI
    - Via
    - Max-Forwards
    - From
    - To
    - Call-ID

- Cseq
- Contact
- Record-Route
- Route
- Supported
- Allow
- P-Preferred-Identity
- Privacy
- Diversion
- Rack
- Required
- RSeq
- Authorization
- Proxy-Authorization
- WWW-Authenticate
- Proxy-Authenticate
- Event
- Refer-To
- Referred-By
- Replaces
- User-Agent
- P-Asserted-ID
- History-Info
- Priority
- Resource-Priority
- Unsupported
- Expires
- Session-Expires
- Min-SE
- Min-Expires

- Bodies:
  - SDP

- DTMF

Configuration:

- 'Action Subject' field: **message**
- 'Action Type' field: **Normalize**

Configuration Examples:

Table 4-14: Normalization Examples

Message Type	Action Subject	Action Type	Description
`invite`	`message`	**Normalize**	Normalizes the entire SIP message (headers and SDP) in INVITE messages.
`invite`	`body.sdp`	**Normalize**	Normalizes only the SDP body in INVITE messages.
`invite`	`header.max-forwards`	**Normalize**	Normalizes the Max-Forwards header in INVITE messages.
`invite`	`header.contact.url`	**Normalize**	Normalizes only the URL of the Contact header in INVITE messages.

Dial Plan Tags

This section describes the syntax for using various Dial Plan tags.

Source and Destination Dial Plan Tags

You can use source and destination Dial Plan tags as conditions ('Condition' field) and values ('Action Value' field) in Message Manipulation rules.

Syntax:

- Source Tag:

```
srctags
srctags.<tag name>
```

- Destination Tag:

```
dsttags
dsttags.<tag name>
```

Applicable fields:

- 'Condition'
- 'Action Value'

Tags cannot be modified by Message Manipulation rules.

Table 4-15: Source and Destination Tags Examples

Message Type	Condition	Action Subject	Action Type	Action Value	Description
`invite`	`srctags=='ny'`	`header.City`	**Add**	`srctags`	If the source tag associated with the call equals "ny", add a header called "City:" with the value set to "ny"

Call Transferor (ReferredBy) Dial Plan Tags

You can use Dial Plan tags of call parties that initiate (transferor) call transfers as conditions ('Condition' field) or values ('Action Value' field) in Call Setup Rules.

Syntax:

```
ReferredByTags.<tag>
```

Applicable fields in the Call Setup Rules table:

- 'Condition'
- 'Action Value'

Tags cannot be modified by Message Manipulation rules.

Table 4-16: Referred By Tags Examples

Condition	Action Subject	Action Type	Action Value	Description
`ReferredByTags.Type == 'PBX'`	`Param.Call.Dst.Numer`	**Add Prefix**	`'+9723976'`	If the leg that sent the SIP REFER request (transferor) is defined with the tag "Type=PBX", the prefix "+9723976" is added to an extension-only destination number (e.g., 4000) of the transferred call.

Using 'cac' Tag for Maximum Concurrent Calls per User

You can configure a different maximum concurrent call limit (incoming or outgoing) for each user. This is done using tags in Call Setup Rules and Dial Plans, which are assigned to the relevant SIP Interface, IP Group, or IP-to-IP Routing rule.

This feature uses the device's hardcoded tag called `cac`. This tag has the format *key*|*value*, where *key* identifies the call (SIP URL user-part or IP address) and *value* defines maximum concurrent call. For example, 'cac=+12345678|3' (or 'cac=10.4.4.4|3') limits the user (or IP address) with phone number +12345678 (or IP address 10.4.4.4) to a maximum of three concurrent calls.

This feature is applicable only to the SBC application.

The following procedure describes how to configure maximum concurrent incoming calls for each specific users.

➢ **To configure maximum concurrent calls for each user:**

1. Configure a Dial Plan with multiple rules, where each rule defines a user by prefix number, and the maximum concurrent calls value by any user-defined tag. For example:
   - 'Name': MyMaxCall
   - 'Prefix': +123
   - 'Tag': cacMax=3
2. Configure a Call Setup Rule to check the existence and value of your Dial Plan tag. For example:
   - 'Condition': SrcTags.cacMax exists And SrcTags.cac !exists
   - 'Action Subject': SrcTags.cac
   - 'Action Type': Modify
   - 'Action Value': Header.From.URL.User + '|' + SrcTags.cacMax
3. Assign the Dial Plan and Call Setup Rule to the relevant SIP Interface, IP Group, or IP-to-IP Routing rule.

If you want all users to have the **same** maximum concurrent calls, you only need to use a Call Setup Rule. For example, the below applies a maximum concurrent incoming call value of 3 to every user (defined by SIP user-part or IP address):

- 'Action Subject': SrcTags.cac
- 'Action Type': Modify
- 'Action Value': (one of following)
  - Per IP address: Param.Message.Address.Src.IP + '|3'
  - Per user: Header.From.Url.User + '|3'

To view a list of 'cac' keys with their current concurrent calls out of their maximum allowed concurrent calls, use the following CLI command:

```
show voip tags-cac
```

To view the above output for a specific user, use the following command:

```
show voip tags-cac key <SIP user part or IP Address>
```

If the maximum number of concurrent calls is reached, the device rejects new calls and sends the following syslog message:

- For incoming calls: "RELEASE_BECAUSE_IN_PER_KEY_CAC_LIMIT_REACHED"

- for outgoing calls: "RELEASE_BECAUSE_OUT_PER_KEY_CAC_LIMIT_REACHED"

ENUM Queries

You can use Call Setup rules to query an ENUM server and to handle responses from ENUM server. ENUM translates ordinary telephone numbers (E.164 telephone numbers) into Internet addresses (SIP URIs), using the ENUM's DNS NAPTR records. Once resolved into a URI, the device can replace the telephone number in the Request-URI of the SIP message with the URI. When configuring Call Setup rules for ENUM queries, configure the 'Request Type' parameter to ENUM.

Syntax:

- Result (i.e., URI) of the ENUM query:

  ```
  enum.result.url.<x>
  ```

  Where x is optional and can be any of the following:

  - type
  - host
  - mhost
  - userphone
  - looseroute
  - bnce
  - cause
  - user
  - transport-type
  - ac-int
  - param

- If ENUM query succeeded or not:

  ```
  enum.found exists
  enum.found !exists
  ```

Applicable table:

- Call Setup Rules table:
  - 'Condition'
  - 'Action Subject' (only enum.result.url)
  - 'Action Value' (only enum.result.url)

Example:

Table 4-17: ENUM Query Example

Request Type	Request Key	Condition	Action Subject	Action Type	Action Value	Description
ENUM	param.call.dst.user	enum.found exists	header.request-uri.url	**Modify**	enum.result.url	Performs an ENUM query using the called number and if successful, replaces the entire SIP Request-URI with the retrieved URI.
ENUM	param.call.dst.user	enum.found exists	header.request-uri.url.user	**Modify**	enum.result.url.user	Performs an ENUM query using the called number and if successful, replac

Request Type	Request Key	Condition	Action Subject	Action Type	Action Value	Description
						es SIP Request-URI user part with the retrieved URI user part

Call Key Variable for REST-Triggered SIPREC

For REST-triggered SIPREC, the device identifies the call to record by using a "Call Key". If the call key value in the incoming REST message is the same as the call key value of the incoming INVITE message (and matched to a rule in the SIP Recording Rules table), the device records the call.

The call key value is obtained from the INVITE message using a Message Manipulation rule that uses one of the following syntax variables in the Action Subject field:

Syntax:

```
Param.Call.HashKey
Param.Peer-Call.HashKey
```

Each variable represents a specific leg of the call. For more details, refer to the device's User's Manual.

The Message Manipulation rule is used internally only (to store the "Call Key" value) and doesn't modify the outgoing INVITE message.

Table 4-18: Call Key Example

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite.request	header.X-ConversationId exists	param.call.hashkey	**Modify**	header.X-ConversationId	Obtains the call key value from the SIP header "X-ConversationId"

SIP URIs and LDAP Queries for Microsoft Skype Presence Feature

When the device is configured to interwork with Microsoft Skype for Business presence feature for third-party endpoints (non-Microsoft endpoints), the device needs to query the LDAP server with the calling and/or called numbers of the third-party endpoints and then retrieve the corresponding SIP URIs of the Skype users. This is the URI that the device uses as the Request-URI in the PUBLISH message that it sends to the Skype for Business Server to indicate change of presence.

The device uses the following syntax in Call Setup Rules in the Action Subject field for these SIP URIs:

Syntax:

- Source SIP URI:

```
presence.src
```

- Destination SIP URI:

```
presence.dst
```

For example, to search for a called mobile number, the searched LDAP Attribute is "mobile" set to the value of the destination number ('mobile=+' + param.call.dst.user). If the entry exists, the query searches for the Attribute userPrincipalName where the SIP URI is defined for the corresponding mobile user. If found, the query returns the Attribute value (i.e., URI) to the device (instructed using the special 'Condition' string "presence.dst" or "presence.src").

Table 4-19: Source and Destination SIP URIs for Skype for Business Presence

Request Type	Request Key	Attributes To Get	Condition	Action Subject	Action Type	Action Value
LDAP	'mobile=+' + param.call.dst.user	userPrincipalName	ldap.attr.mobile exists	presence.dst	Add	ldap.attr.user PrincipalName
LDAP	'mobile=+' + param.call.src.user	userPrincipalName	ldap.attr.mobile exists	presence.src	Add	ldap.attr.user PrincipalName

HTTP POST and GET Requests

You can use Call Setup Rules to query HTTP-based servers using the HTTP GET and HTTP POST request methods. The response from the HTTP server can be used for various functionality such as routing, or its data can be saved, for example, as a call/session variable to use in SIP message manipulations.

You can also use Call Setup Rules to notify the server of a specific condition, using HTTP POST notifications. In this case, the Call Setup Rule does not expect a response from the server for these HTTP message notifications.

Syntax:

- To refer to an HTTP response code received from the HTTP server:

  ```
  Http.Response.Status
  ```

  This syntax is used in the 'Condition' field.

- To refer to the body in the HTTP response (string after the HTTP headers):

  ```
  Http.Response.Body
  ```

  This syntax can be used in the following fields:

  - 'Request Key'
  - 'Condition'
  - 'Action Value'

- To refer to a condition if an HTTP response exists:

```
Http.Found
```

This syntax is used in the 'Condition' field.

- To refer to the body in the sent HTTP POST request:

```
Http.Request.Body
```

This syntax can be used in the following fields:

- 'Condition'
- 'Action Subject' (with 'Action Type' configured to Modify for changing the body value entirely, or Add for appending and concatenating the new body value to the existing body)
- 'Action Value'

- To refer to the Content-Type header in the sent HTTP request:

```
Http.Request.Content-Type
```

This syntax can be used in the following fields:

- 'Condition'
- 'Action Subject' (with 'Action Type' configured to Modify)
- 'Action Value'

For POST requests, the header is omitted by default; for GET requests, it is set to "html/text". Commonly used Content-Type values include "application/json", "application/octet-stream", "message/http", "html/text", and "application/x-www-form-urlencoded".

The HTTP server is configured as a Remote Web Service in the Remote Web Services table with the 'Type' parameter configured to General. The 'Request Type' parameter in the Call Setup Rules table must be configured to HTTP GET, HTTP POST Query, or HTTP POST Notification and the 'Request Target' to the name of the Remote Web Service (case-sensitive).

- When the Call Setup Rule doesn't need to do any action after the HTTP request is sent (e.g., for HTTP POST notification requests), you can use the value None in the 'Action Type' field.
- Unlike HTTP GET requests which include all required data in the URL, HTTP POST requests typically include a URL and a message body.

Table 4-20: Examples of HTTP GET and POST Requests

Request Type	Request Target	Request Key	Condition	Action Subject	ActionType	Action Value	Description
Example 1							
HTTP GET	MyHTTP-Server	'?user='+param.call.src.user		param.call.src.name	Modify	http.response.body	Searches the server for the caller's username and then modifies the From

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							header (caller ID) in the outgoing SIP message, by adding the value (username) obtaine

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
Example 2							
HTTPGET	MyHTTP-Server	'?user='+param.call.src.user	http.response.status=='200' and http.response.body regex <value>)(.*)(</value>)	header.X-Info	Add	$2	Searches the server for the caller's username and if the HTTP respon

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							se co de is 20 0 O K, it th en ad ds th e "X - Inf o" he ad er wi th th e val ue ob tai ne d fr o m th

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							), in the outgoing SIP message.
Example 3							
None			`Param.Call.Dst.User != '911'`		Exit	`True`	If the destination number is not 911, then exit th

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							eCallSetupRulestable.
None				Http.Request.Content-Type	Modify	'application/json'	Changes the HTTPGET Request's Content-Type header value

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							to the string value configured in the 'ActionValue' field.
None				Http.Request.Body	Modify	'EmergencyCaller='+param.call.src.user	Changes the HTTP GET requ es

Request Type	Request Target	Request Key	Condition	Action Subject	ActionType	Action Value	Description
							t's body string value to the string value configured in the 'Action Value' field.
HTTP POST Notificat	MyHTTP-	'emergencyNotifier'			None		Sends an HTTP

RequestType	RequestTarget	Request Key	Condition	Action Subject	ActionType	Action Value	Description
ion	Server						POST request to notify the HTTP server.

5 Typical Examples

The following table provides a summary of typical examples of Message Manipulation rules.

Table 5-1: Message Manipulation Examples

Message Type	Condition	Action Subject	Action Type	Action Value	Description
invite.request	param.message.sdp.ip=='flowers.com'	header.diversion	Add	'<sip:WeSellFlowers@p4.isp.com>;reason=time-of-day'	In INVITE requests, add a Diversion header if the c line in the SDP is set to "flowers.com".
info.response	header.request-uri.methodtype=='488'	header.request-uri.methodtype	Modify	'503'	Change the Request-URI method

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					type to 503 from 488 in INFO response messages
info.response.180		header.request-uri.methodtype	**Modify**	'183'	Change request type method to 183 in 180 response messages.
invite.request	header.expires.time < '88888'	header.organisation	**Add**	'audiocodes'	Check the time parameter in Expires

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					headers. If it is less than 88888, add an organization header to the INVITE request message.
register.request		header.contact.param.newparam	Add	'newValue'	Add new Param with a value of new Value as a gene

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					ral header level param to REGISTER Contact headers
subscribe.response		header.remote-party-id.0.partytype	**Modify**	'2'	In Subscribe response messages, change the party type to 'called' (note, 1="calling", 2="called",

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					3="redirect") in the 1st Remote-Party-ID header.
invite.response		header.from.param.hello	**Remove**		Remove the param named "hello" from From headers in INVITE responses.
any		header.user-agent	**Modify**	'TelcoA'	Change the User-

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					Agent header to telco A.
any		header.from.quotecontrol	**Modify**	'0'	Removes quotation marks surrounding display name in From header.
any		header.user-to-user.param.purpose	**Add**	'isdn-network'	Adds the parameter "purpose" with value "isd

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					n-network' to the User-to-User header.
any	header.to.url.user num> '20'	header.Result	Add	'user part is numerically greater than 20'	If the user part in the To header is numerically greater than 20, the "Result" header is added to the

Message Type	Condition	Action Subject	Action Type	Action Value	Description
					message with the value "user part is numerically greater than 20".

6 Message Manipulation Syntax Reference

This appendix provides a detailed description on the support and syntax for configuring SIP message manipulation rules.

Action Type

The actions that can be done on SIP message manipulation are listed in the following table.

Table 6-1: Action Types

Action	Value
Add	0
Remove	1
Modify	2
Add Prefix	3
Add Suffix	4
Remove Suffix	5
Remove Prefix	6

The maximum length of the value for a manipulation is 299 characters.

Header Types

Accept

An example of the header is shown below:

```
Accept: application/sdp
```

Multiple header fields support: No

The header properties are shown in the following table:

Header Level Action	Action Type			
	Add	**Delete**	**Modify**	**List Entries**

Header Level Action	Action Type			
Operations Supported	Yes	Yes	No	N/A
Keyword	**Sub Types**		**Attributes**	
N/A	N/A		N/A	

Below is a header manipulation example:

Rule:	If the supported header does not contain 'mm,100rel,timer,replaces', then in all INVITE messages add an Accept header: `MessageManipulations 8 = 1, invite, header.supported != 'mm,100rel,timer,replaces', header.accept, 0, ' application/x-private ', 0;`
Result:	Accept: application/x-private

Accept-Language

An example of the header is shown below:

```
Accept-Language: da, en-gb;q=0.8, en;q=0.7
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	No	N/A
Keyword	**Sub Types**		**Attributes**	
N/A	N/A		N/A	

Below is a header manipulation example:

Rule:	Add a new Language header to all INVITE messages: `MessageManipulations 0 = 1, invite, , header.accept-language, 0, 'en, il, cz, it', 0;`
Result:	Accept-Language: en, il, cz, it

Allow

An example of the header is shown below:

```
Allow:
REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFER,INFO,SUBSCRIBE
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	No	N/A

Keyword	Sub Types	Attributes
N/A	N/A	Read/Write

Below is a header manipulation example:

Rule:	Add an Allow header to all INVITE messages: `MessageManipulations 0 = 1, invite, , header.allow, 0, 'REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK,REFER,INFO,SUBSCRIBE, XMESSAGE', 0;`
Result:	Allow: REGISTER,OPTIONS,INVITE,ACK,CANCEL,BYE,NOTIFY,PRACK, REFER,INFO,SUBSCRIBE, XMESSAGE

Call-Id

An example of the header is shown below:

```
Call-ID: JNIYXOLCAIWTRHWOINNR@10.132.10.128
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	No	NA

Keyword	Sub Types	Attributes
ID	String	Read Only

Below is a header manipulation example:

Rule:	Add a proprietary header to all INVITE messages using the data in the Call-id header: `MessageManipulations 0 = 1, invite, , header.Xitsp-abc, 0, header.call-id, 0;`
Result:	Xitsp-abc: GIAPOFWRBQKJVAETIODI@10.132.10.128

Contact

An example of the header is shown below:

Contact: <sip:555@10.132.10.128:5080>

Multiple header fields support: Yes

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	No	3

Keyword	Sub Types	Attributes
Expires	Integer	Read/Write
GruuContact	String	Read/Write
IsGRUU	Boolean	Read/Write
Name	String	Read/Write
Param	Param	Read/Write
URL	URL on page 173	Read/Write*

* Host name cannot be modified in the URL structure for a contact header.

Below is a header manipulation example:

Rule:	Change the user part in the Contact header in all INVITE messages to fred: `MessageManipulations 0 = 1, Invite, ,header.contact.url.user, 2, 'fred', 0;`

Result:	Contact: <sip:fred@10.132.10.128:5070>

Cseq

An example of the header is shown below:

```
CSeq: 1 INVITE
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	No	N/A

Keyword	Sub Types	Attributes
Num	Integer	Read Only
Type	String	Read Only

Below is a header manipulation example:

Rule:	If the Cseq number is 1, then modify the user in the Contact header to fred. `MessageManipulations 0 = 1, Invite, header.cseq.num=='1',header.contact.url.user, 2, 'fred', 0;`
Result:	Contact: <sip:fred@10.132.10.128:5070>

Diversion

An example of the header is shown below:

```
Diversion: <sip:654@IPG2Host;user=phone>;reason=user-busy;screen=no;privacy=off;counter=1
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	3

Keyword	Sub Types	Attributes
Name	String	Read/Write
Param	Param	Read/Write
Privacy	Enum Privacy (see Privacy on page 177)	Read/Write
Reason	Enum Reason (see Reason (Diversion) on page 177)	Read/Write
Screen	Enum Screen (see Screen on page 182)	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Diversion header to all INVITE messages: `MessageManipulations 0 = 1, invite, , header.Diversion, 0, '<tel:+101>;reason=unknown; counter=1;screen=no; privacy=off', 0;`
	Result:	Diversion: <tel:+101>;reason=user-busy;screen=no;privacy=off;counter=1
Example 2	Rule:	Modify the Reason parameter in the header to 1, see Reason (Diversion) on page 177 for possible values: `MessageManipulations 1 = 1, invite, , header.Diversion.reason, 2, '1', 0;`
	Result:	Diversion: <tel:+101>;reason=user-busy;screen=no;privacy=off;counter=1
Example 3	Rule:	The URL in the Diversion header is modified to that which is contained in the header URL: `MessageManipulations 2 = 1, invite, , header.Diversion.URL, 2, header.from.url, 0;`
	Result:	Diversion:<sip:555@IPG2Host;user=phone>;reason=user-busy;screen=no;privacy=off;counter=1

Event

An example of the header is shown below:

```
Event: foo; id=1234
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
EventKey	Event Structure (see Event Structure on page 171)	Read/Write
Param	Param	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add parameter itsp-abc=voip to the Event header: `MessageManipulations 0 = 1, invite, , header.event.param.itsp-abc, 0, 'voip' , 0;`
	Result:	`Event: foo;id=1234;itsp-abc=voip`
Example 2	Rule:	Modify the Event ID string: `MessageManipulations 1 = 1, invite, , header.event.EVENTKEY.id, 2, '5678', 0;`
	Result:	`Event: foo;id=5678;`
Example 3	Rule:	Modify the Event package enum: `MessageManipulations 2 = 1, invite, , header.event.EVENTKEY.EVENTPACKAGE, 2, '2', 0;`
	Result:	`Event: refer;id=5678`

From

An example of the header is shown below:

From:
<sip:555@10.132.10.128;user=phone>;tag=YQLQHCAAYBWKKRVIMWEQ

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	Yes	NA

Keyword	Sub Types	Attributes
Name	String	Read/Write
Param	Param	Read/Write
tag	String	Read Only
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Change the user part of the From header if the user is not 654: `MessageManipulations 8 = 1, invite, header.from.url.user != '654', header.from.url.user, 2, 'fred', 0;`
	Result:	From: <sip:fred@IPG2Host;user=phone>;tag=1c20161
Example 2	Rule:	Add a new parameter to the From header called p1 and set its value to myParameter: `MessageManipulations 1 = 1, Invite.request, ,header.from.param.p1, 0, 'myParameter', 0;`
	Result:	From: <sip:fred@IPG2Host;user=phone>;p1=myParameter;tag=1c5891
Example 3	Rule:	Modify the URL in the From header: `MessageManipulations 0 = 1, any, , header.from.url, 2, 'sip:3200@110.18.5.41;tusunami=0', 0;`

	Result:	From: <sip:3200@110.18.5.41;user=phone;tusunami=0>;tag=1c23750

History-Info

An example of the header is shown below:

```
History-Info: <sip:UserA@ims.example.com;index=1>
History-Info: <sip:UserA@audc.example.com;index=2>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	20

Keyword	Sub Types	Attributes
HistoryInfo	String	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a new History-Info header to the message: `MessageManipulations 0 = 1, any, , header.History-Info, 0, '<sip:UserA@audc.mydomain.com;index=3>', 0`
	Result:	History-Info:sip:UserA@ims.example.com;index=1 History-Info:sip:UserA@audc.example.com;index=2 History-Info: <sip:UserA@audc.mydomain.com;index=3>
Example 2	Rule:	Delete an unwanted History-Info header from the message: `MessageManipulations 0 = 1, any, , header.History-Info.1, 1, , 0;`
	Result:	History-Info: <sip:UserA@ims.example.com;index=1>
Example 3	Rule:	Delete all History-Info from the message:

		`MessageManipulations 0 = 1, any, , header.History-Info, 1, , 0;`
	Result:	All history-info headers are removed.

Min-Se and Min-Expires

An example of the header is shown below:

```
Min-SE: 3600
Min-Expires: 60
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Param	Param	Read/Write
Time	Integer	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Min-Se header to the message using a value of 50: `MessageManipulations 1 = 1, any, , header.min-se, 0, '50', 0;`
	Result:	Min-SE: 50
Example 2	Rule:	Modify a Min-Expires header with the min-expires value and add an additional 0: `MessageManipulations 0 = 1, Invite, , header.Min-Expires.param, 2, header.Min-Expires.time + '0', 0;`
	Result:	Min-Expires: 340;3400
Example 3	Rule:	Modify a Min-Expires header changing the time to 700:

		`MessageManipulations 0 = 1, Invite, , header.Min-Expires.time, 2, '700', 0;`
	Result:	Min-Expires: 700

P-Asserted-Identity

An example of the header is shown below:

```
P-Asserted-Identity: Jane Doe <sip:567@itsp.com>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	2

Keyword	Sub Types	Attributes
URL	URL Structure (see URL on page 173)	Read/Write
Name	String	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a P-Asserted-Id header to all INVITE messages: `MessageManipulations 2 = 1, invite, , header.p-asserted-identity, 0, '<sip:567@itsp.com>', 0;`
	Result:	P-Asserted-Identity: <sip:567@itsp.com>
Example 2	Rule:	Modify the P-Asserted-Identity host name to be the same as the host name in the To header: `MessageManipulations 2 = 1, invite, , header.p-asserted-identity.URL.host, 2, header.to.url.host, 0;`
	Result:	P-Asserted-Identity: <sip:567@10.132.10.128>

P-Associated-Uri

An example of the header is shown below:

```
P-Associated-URI: <sip:12345678@itsp.com>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	2

Keyword	Sub Types	Attributes
Name	String	Read/Write
Param	Param	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a P-Associated-Uri header to all INVITE response messages: `MessageManipulations 5 = 1, register.response, ,header.P-Associated-URI, 0, '<sip:admin@10.132.10.108>', 0;`
	Result:	`P-Associated-URI:<sip:admin@10.132.10.108>`
Example 2	Rule:	Modify the user portion of the URL in the header to 'alice': `MessageManipulations 5 = 1, register.response, ,header.P-Associated-URI.url.user, 2, 'alice', 0;`
	Result:	`P-Associated-URI:<sip:alice@10.132.10.108>`

P-Called-Party-Id

An example of the header is shown below:

```
P-Called-Party-ID: <sip:2000@gw.itsp.com>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Name	String	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a P-Called-Party-Id header to all messages: `MessageManipulations 8 = 1, any, , header.p-called-party-id, 0, 'sip:2000@MSBG.ITSP.COM', 0;`
	Result:	P-Called-Party-ID: <sip:2000@gw.itsp.com>
Example 2	Rule:	Append a parameter (p1) to all P-Called-Party-Id headers: `MessageManipulations 9 = 1, invite, , header.p-called-party-id.param.p1, 0, 'red', 0;`
	Result:	P-Called-Party-ID: <sip:2000@gw.itsp.com>;p1=red
Example 3	Rule:	Add a display name to the P-Called-Party-Id header: `MessageManipulations 3 = 1, any, , header.p-called-party-id.name, 2, 'Secretary', 0;`
	Result:	P-Called-Party-ID: Secretary <sip:2000@gw.itsp.com>;p1=red

P-Charging-Vector

An example of the header is shown below:

```
P-Charging-Vector: icid-value=1234bc9876e; icid-generated-at=192.0.6.8; orig-ioi=home1.net
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	No	N/A

Keyword	Sub Types	Attributes
N/A	N/A	N/A

Below are header manipulation examples:

Rule:	Add a P-Charging-Vector header to all messages: `MessageManipulations 1 = 1, any, , header.P-Charging-Vector, 0, 'icid-value=1234bc9876e; icid-generated-at=192.0.6.8; orig-ioi=home1.net', 0;`
Result:	P-Charging-Vector: icid-value=1234bc9876e; icid-generated-at=192.0.6.8; orig-ioi=home1.net

P-Preferred-Identity

An example of the header is shown below:

P-Preferred-Identity: "Cullen Jennings" <sip:fluffy@abc.com>

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	2

Keyword	Sub Types	Attributes
Name	String	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a P-Preferred-Identity header to all messages: `MessageManipulations 1 = 1, any, , header.P-Preferred-Identity, 0, 'Cullen Jennings <sip:fluffy@abc.com>', 0;`

	Result:	P-Preferred-Identity: "Cullen Jennings" <sip:fluffy@abc.com>
Example 2	Rule:	Modify the display name in the P-Preferred-Identity header: `MessageManipulations 2 = 1, any, , header.P-Preferred-Identity.name, 2, 'Alice Biloxi', 0;`
	Result:	P-Preferred-Identity: "Alice Biloxi" <sip:fluffy@abc.com>

Privacy

An example of the header is shown below:

```
Privacy: none
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	No	N/A

Keyword	Sub Types	Attributes
privacy	Privacy Struct on page 172	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a privacy header and set it to ''session': `MessageManipulations 1 = 1, any, , header.Privacy, 0, 'session', 0;`
	Result:	Privacy: session
Example 2	Rule:	Add 'user' to the list: `MessageManipulations 1 = 3, , , header.privacy.privacy.user, 2, '1', 0;`

	Result:	Privacy: session;user

Proxy-Require

An example of the header is shown below:

```
Proxy-Require: sec-agree
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Capabilities	SIPCapabilities on page 172	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Proxy-Require header to the message: `MessageManipulations 1 = 1, any, , header.Proxy-Require, 0, 'sec-agree', 0;`
	Result:	Proxy-Require: sec-agree
Example 2	Rule:	Modify the Proxy-Require header to itsp.com: `MessageManipulations 2 = 1, any, , header.Proxy-Require, 2, 'itsp.com' , 0;`
	Result:	Proxy-Require: itsp.com
Example 3	Rule:	Set the privacy options tag in the Proxy-Require header: `MessageManipulations 0 = 0, invite, , header.Proxy-Require.capabilities.privacy, 0, 1 , 0;`
	Result:	Proxy-Require: itsp.com, privacy

Reason

An example of the header is shown below:

```
Reason: SIP ;cause=200 ;text="Call completed elsewhere"
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
MLPP	MLPP Structure (see MLPP on page 171)	Read/Write
Reason	Reason Structure (see Reason Structure on page 172)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Reason header: `MessageManipulations 0 = 1, any, ,header.reason, 0, 'SIP;cause=200;text="Call completed elsewhere"', 0;`
	Result:	Reason: SIP ;cause=200 ;text="Call completed elsewhere"
Example 2	Rule:	Modify the reason cause number: `MessageManipulations 0 = 1, any, ,header.reason.reason.cause, 0, '200', 0;`
	Result:	Reason: Q.850 ;cause=180 ;text="Call completed elsewhere"
Example 3	Rule:	Modify the cause number: `MessageManipulations 0 = 1, any, ,header.reason.reason.reason, 0, '483', 0;`
	Result:	Reason: SIP ;cause=483 ;text="483 Too Many Hops"

The protocol (SIP or Q.850) is controlled by setting the cause number to be greater than 0. If the cause is 0, then the text string (see Example 3) is generated from the reason number.

Referred-By

An example of the header is shown below:

```
Referred-By: <sip:referrer@referrer.example>;
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
param	param	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Referred-By header: `MessageManipulations 0 = 1, any, ,header.Referred-By, 0, '<sip:refer@refer.com>', 0;`
	Result:	`Referred-By: <sip: sip:refer@refer.com>`
Example 2	Rule:	Modify the host: `MessageManipulations 0 = 1, any, ,header.Referred-By.url.host, 0, 'yahoo.com', 0;`
	Result:	`Referred-By: <sip:refer@yahoo.com>`
Example 3	Rule:	Add a new parameter to the header: `MessageManipulations 0 = 1, any, ,header.Referred-By.param.p1, 0, 'fxs', 0`

	Result:	Referred-By: <sip:referrer@yahoo.com>;p1=fxs

Refer-To

An example of the header is shown below:

```
Refer-To: sip:conference1@example.com
Refer-To:
<sips:a8342043f@atlanta.example.com?Replaces=12345601%40atlanta.example.com%3bfrom-tag%3d314159%3bto-tag%3d1234567>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	No	N/A

Keyword	Sub Types	Attributes
N/A	N/A	N/A

Below are header manipulation examples:

Example 1	Rule:	Add a basic header: `MessageManipulations 0 = 1, any, ,header.Refer-to, 0, '<sip:referto@referto.com>', 0;`
	Result:	Refer-To: <sip:referto@referto.com>
Example 2	Rule:	Add a Refer-To header with URI headers: `MessageManipulations 0 = 1, any, ,header.Refer-to, 0, '<sips:a8342043f@atlanta.example.com?Replaces=12345601%40atlanta.example.com%3bfrom-tag%3d314159%3bto-tag%3d1234567>', 0;`
	Result:	Refer-To: <sips:a8342043f@atlanta.example.com?Replaces=12345601%40atlanta.example.com%3bfrom-tag%3d314159%3bto-tag%3d1234567>

Remote-Party-Id

An example of the header is shown below:

```
Remote-Party-ID: "John Smith" <sip:john.smith@itsp.com>;party=calling;
privacy=full;screen=yes
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	3

Keyword	Sub Types	Attributes
Counter	Integer	Read/Write
Name	String	Read/Write
NumberPlan	Enum Number Plan (see Number Plan on page 176)	Read/Write
NumberType	Enum Number Type (see Number Type on page 177)	Read/Write
Param	Param	Read/Write
Privacy	Enum Privacy (see Privacy on page 177)	Read/Write
Reason	Enum Reason (RPI) (see Reason (Remote-Party-Id) on page 181)	Read/Write
Screen	Enum Screen (see Screen on page 182)	Read/Write
ScreenInd	Enum ScreenInd (see ScreenInd on page 182)	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Remote-Party-Id header to the message: `MessageManipulations 0 = 1, invite, ,header.REMOTE-PARTY-ID, 0, '<sip:999@10.132.10.108>;party=calling', 0;`
	Result:	Remote-Party-ID: <sip:999@10.132.10.108>;party=calling;npi=0;ton=0

Example 2	Rule:	Create a Remote-Party-Id header using the url in the From header using the + operator to concatenate strings: `MessageManipulations 0 = 1, Invite, ,header.REMOTE-PARTY-ID, 0, '<'+header.from.url +'>' + ';party=calling', 0;`
	Result:	Remote-Party-ID: <sip:555@10.132.10.128;user=phone>;party=calling;npi=0;ton=0
Example 3	Rule:	Modify the number plan to 1 (ISDN): `MessageManipulations 1 = 1, invite, , header.Remote-Party-ID.numberplan, 2, '1', 0;`
	Result:	Remote-Party-ID: <sip:555@10.132.10.128;user=phone>;party=calling;npi=1;ton=0
Example 4	Rule:	Modify the Remote-Party-Id header to set the privacy parameter to 1 (Full): `MessageManipulations 1 = 1, invite, , header.Remote-Party-ID.privacy, 2, '1', 0;`
	Result:	Remote-Party-ID: <sip:555@10.132.10.128;user=phone>;party=calling;privacy=full;npi=0;ton=0

Request-Uri

An example of the header is shown below:

```
sip:alice:secretword@atlanta.com;transport=tcp
SIP/2.0 486 Busy Here
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	Yes	NA

Keyword	Sub Types	Attributes
Method	String	Read/Write
MethodType	Enum: ■ 5: INVITE ■ 7: BYE ■ 8: OPTIONS ■ 9: ACK ■ 10: CANCEL ■ 11: REGISTER ■ 12: INFO ■ 13: MESSAGE ■ 14: NOTIFY ■ 15: REFER ■ 16: SUBSCRIBE ■ 17: PRACK ■ 18: UPDATE ■ 19: PUBLISH ■ 21: SERVICE	Read/Write
URI	String	Read/Write
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Test the Request-URI transport type. If 1 (TCP), then modify the URL portion of the From header: `MessageManipulations 1 = 1, Invite.request, header.REQUEST-URI.url.user == '101', header.REMOTE-PARTY-ID.url, 2, 'sip:3200@110.18.5.41;tusunami=0', 0;`
	Result:	Remote-Party-ID: <sip:3200@110.18.5.41;tusunami=0>;party=calling;npi=0;ton=0

Example 2	Rule:	If the method type is 5 (INVITE), then modify the Remote-Party-Id header: `MessageManipulations 2 = 1, Invite.request, header.REQUEST-URI.methodtype == '5', header.REMOTE-PARTY-ID.url, 2, 'sip:3200@110.18.5.41;tusunami=0', 0;`
	Result:	Remote-Party-ID: <sip:3200@110.18.5.41;tusunami=0>;party=calling;npi=0;ton=0
Example 3	Rule:	For all request URI's whose method types are 488, modify the message type to a 486: `MessageManipulations 1 = 1, , header.request-uri.methodtype=='488', header.request-uri.methodtype, 2, '486', 0;`
	Result:	SIP/2.0 486 Busy Here

Require

An example of the header is shown below:

```
Require: 100rel
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Capabilities	SIPCapabilities on page 172 Struct	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Require header to all messages: `MessageManipulations 1 = 1, , ,header.require, 0, 'early-session,em,replaces', 0;`

	Result:	Require: em,replaces,early-session
Example 2	Rule:	If a Require header exists, then delete it: `MessageManipulations 2 = 1, Invite, header.require exists ,header.require, 1, '', 0;`
	Result:	The Require header is deleted.
Example 3	Rule:	Set the early media options tag in the header: `MessageManipulations 0 = 0, invite, , header.require.capabilities.earlymedia, 0, 1 , 0;`
	Result:	Require: em,replaces,early-session, early-media
Example 4	Rule:	Set the privacy options tag in the Require header: `MessageManipulations 0 = 0, invite, , header.require.capabilities.privacy, 0, 1 , 0;`
	Result:	Require: em,replaces,early-session, privacy

Resource-Priority

An example of the header is shown below:

```
Resource-Priority: wps.3
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	2

Keyword	Sub Types	Attributes
Namespace	String	Read/Write
RPriority	String	Read/Write

Retry-After

An example of the header is shown below:

```
Retry-After: 18000
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Time	Integer	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Retry-After header: `MessageManipulations 2 = 1, Invite, ,header.Retry-After, 0, '3600', 0;`
	Result:	`Retry-After: 3600`
Example 2	Rule:	Modify the Retry-Time in the header to 1800: `MessageManipulations 3 = 1, Invite, ,header.Retry-After.time, 2, '1800', 0;`
	Result:	`Retry-After: 1800`

Server or User-Agent

An example of the header is shown below:

```
User-Agent: Sip Message Generator V1.0.0.5
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
N/A	N/A	N/A

Below are header manipulation examples:

Example 1	Rule:	Remove the User-Agent header: `MessageManipulations 2 = 1, Invite, ,header.user-agent, 1, '', 0;`
	Result:	The header is removed.
Example 2	Rule:	Change the user agent name in the header: `MessageManipulations 3 = 1, Invite, ,header.user-agent, 2, 'itsp analogue gateway', 0;`
	Result:	User-Agent: itsp analog gateway

Service-Route

An example of the header is shown below:

```
Service-Route: <sip:P2.HOME.EXAMPLE.COM;lr>,
<sip:HSP.HOME.EXAMPLE.COM;lr>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	7

Keyword	Sub Types	Attributes
ServiceRoute	String	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add two Service-Route headers:
		`MessageManipulations 1 = 1, Invite, ,header.service-route, 0, '<P2.HOME.EXAMPLE.COM;lr>', 0;`
		`MessageManipulations 2 = 1, Invite,`

		,header.service-route, 0, '<sip:HSP.HOME.EXAMPLE.COM;lr>', 0;
	Result:	Service-Route:<P2.HOME.EXAMPLE.COM;lr> Service-Route: <sip:HSP.HOME.EXAMPLE.COM;lr>
Example 2	Rule:	Modify the Service-Route header in list entry 1: MessageManipulations 3 = 1, Invite, ,header.service-route.1.serviceroute, 2, '<sip:itsp.com;lr>', 0;
	Result:	Service-Route:sip:itsp.com;lr Service-Route: <sip:HSP.HOME.EXAMPLE.COM;lr>
Example 3	Rule:	Modify the Service-Route header in list entry 0: MessageManipulations 4 = 1, Invite, ,header.service-route.0.serviceroute, 2, '<sip:home.itsp.com;lr>', 0;
	Result:	Service-Route:sip:home.itsp.com;lr Service-Route: <sip:itsp.com;lr>

Session-Expires

An example of the header is shown below:

```
Session-Expires: 480
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Param	Param	Read/Write
Refresher	Enum Refresher (see Refresher on page 181)	Read/Write
Time	Integer	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add a Session-Expires header: `MessageManipulations 0 = 1, any, , header.Session-Expires, 0, '48' + '0', 0;`
	Result:	Session-Expires: 480
Example 2	Rule:	Modify the Session-Expires header to 300: `MessageManipulations 1 = 1, any, , header.Session-Expires.time, 2, '300', 0;`
	Result:	Session-Expires: 300
Example 3	Rule:	Add a param called longtimer to the header: `MessageManipulations 1 = 1, any, , header.Session-Expires.param.longtimer, 0, '5', 0;`
	Result:	Session-Expires: 480;longtimer=5
Example 4	Rule:	Set the refresher to 1 (UAC): `MessageManipulations 3 = 1, any, , header.session-expires.refresher, 2, '1', 0;`
	Result:	Session-Expires: 300;refresher=uac;longtimer=5

Subject

An example of the header is shown below:

Subject: A tornado is heading our way!

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Subject	String	Read/Write

Below is a header manipulation example:

Rule:	Add a Subject header: `MessageManipulations 0 = 1, any, , header.Subject, 0, 'A tornado is heading our way!', 0;`
Result:	Subject: A tornado is heading our way!

Supported

An example of the header is shown below:

```
Supported: early-session
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Capabilities	SIPCapabilities Struct	Read/Write

Below is a header manipulation example:

Example 1	Rule:	Add a Supported header: `MessageManipulations 1 = 1, Invite, ,header.supported, 0, 'early-session', 0;`
	Result:	Supported: early-session
Example 2	Rule:	Set path in the Supported headers options tag: `MessageManipulations 0 = 0, invite, , header.supported.capabilities.path, 0, true, 0;`

	Result:	Supported: early-session, path

To

An example of the header is shown below:

```
To: <sip:101@10.132.10.128;user=phone>
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	No	NA

Keyword	Sub Types	Attributes
Name	String	Read/Write
Param	Param	Read/Write
tag	String	Read Only
URL	URL Structure (see URL on page 173)	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Set the user phone Boolean to be false in the To header's URL: `MessageManipulations 4 = 1, invite.request, , header.to.url.UserPhone, 2, '0', 0;`
	Result:	To: <sip:101@10.132.10.128>
Example 2	Rule:	Change the URL in the To header: `MessageManipulations 4 = 1, invite.request, , header.to.url.UserPhone, 2, '0', 0;`
	Result:	To: <sip:101@10.20.30.60:65100>
Example 3	Rule:	Set the display name to 'Bob': `MessageManipulations 5 = 1, invite.request, , header.to.name, 2, 'Bob', 0;`

	Result:	To: "Bob D" sip:101@10.20.30.60:65100
Example 4	Rule:	Add a proprietary parameter to all To headers: `MessageManipulations 6 = 1, invite.request, , header.to.param.artist, 0, 'singer', 0;`
	Result:	To: "Bob D" <sip:101@10.20.30.60:65100>;artist=singer

Unsupported

An example of the header is shown below:

```
Unsupported: 100rel
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	N/A

Keyword	Sub Types	Attributes
Capabilities	SIPCapabilities Struct	Read/Write

Below are header manipulation examples:

Example 1	Rule:	Add an Unsupported header to the message: `MessageManipulations 0 = 1, Invite.response, ,header.unsupported, 0, 'early-session, myUnsupportedHeader', 0;`
	Result:	Unsupported: early-session
Example 2	Rule:	Modify the Unsupported header to 'replaces': `MessageManipulations 1 = 1, Invite, ,header.unsupported, 2, 'replaces', 0;`
	Result:	Unsupported: replaces

Example 3	Rule:	Set the path in the Unsupported headers options tag: `MessageManipulations 0 = 0, invite, , header.unsupported.capabilities.path, 0, true, 0;`
	Result:	Unsupported: replaces, path

Via

An example of the header is shown below:

```
Via: SIP/2.0/UDP 10.132.10.128;branch=z9hG4bKUGOKMQPAVFKTAVYDQPTB
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	No	No	No	10

Keyword	Sub Types	Attributes
Alias	Boolean	Read Only
Branch	String	Read Only
Host	Host Structure (see Host on page 171)	Read Only
MAddrIp	gnTIPAddress	Read Only
Param	Param	Read/Write
Port	Integer	Read Only
TransportType	Enum TransportType (see TransportType on page 182)	Read Only

Below is a header manipulation example:

Rule:	Check the transport type in the first Via header and if it's set to UDP, then modify the From header's URL: `MessageManipulations 0 = 1, Invite.request, header.VIA.0.transporttype == '0', header.from.url, 2, 'sip:3200@110.18.5.41;tusunami=0', 0;`

Result:	From: <sip:3200@110.18.5.41;user=phone;tusunami=0>;tag=1c7874

Warning

An example of the header is shown below:

```
Warning: 307 isi.edu "Session parameter 'foo' not understood"
Warning: 301 isi.edu "Incompatible network address type 'E.164'"
```

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	2

Keyword	Sub Types	Attributes
N/A	N/A	N/A

Below is a header manipulation example:

Rule:	Add a Warning header to the message: `MessageManipulations 0 = 1, Invite.response.180, ,header.warning, 0, '399 source.host.com \"Incompatible\"', 0;`
Result:	Warning: 399 source.host.com “Incompatible”

Unknown Header

An Unknown header is a SIP header that is not included in this list of supported headers. An example of the header is shown below:

```
MYEXP: scooby, doo, goo, foo
```

The device's Message Manipulation feature supports up to three occurrences of a given unknown header in a single message. If there are more than three of the same header, the device removes the extras.

The header properties are shown in the following table:

Header Level Action	Add	Delete	Modify	List Entries
Operations Supported	Yes	Yes	Yes	3

Keyword	Sub Types	Attributes
N/A	N/A	N/A

Below are header manipulation examples:

Example 1	Rule:	Add a custom header to all messages: `MessageManipulations 0 = 1, , , header.myExp, 0, 'scooby, doo, goo, foo', 0;`
	Result:	myExp: scooby, doo, goo, foo
Example 2	Rule:	Create a new header called "media", whose value is a concatenation of the time in the Session-Expires header, followed by "000", followed by ";refresher=", followed by "1" or "2", depending on whether the refresher parameter in the Session-Expires header has the value 'UAC' or 'UAS': `MessageManipulations 0 = 1, any, , header.media, 0, header.Session-Expires.time + '000' + ';refresher=' + header.Session-Expires.Refresher, 0;`
	Result:	media: 3600000;refresher=1

Example 3	Rule:	Create lists of Unknown headers:
		`MessageManipulations 1 = 1, Invite, , header.myExp.1, 0, 'scooby, doo, goo, foo1', 0;`
		`MessageManipulations 2 = 1, Invite, , header.myExp.2, 0, 'scooby, doo, goo, foo2', 0;`

	Result:	myExp: scooby, doo, goo, foo1 myExp: scooby, doo, goo, foo2
Example 4	Rule:	Remove the SIP header 'colour' from INVITE messages:

		`MessageManipulations 1 = 1, Invite, , header.colour, 1, '', 0;`
	Result:	The colour header is removed.

Structure Definitions

Event Structure

The Event structure is used in the Event header (see Event on page 143).

Table 6-2: Event Structure

Keyword	Sub Types	Attributes
EventPackage	Enum Event Package (see Event Package on page 175)	Read/Write
EventPackageString*	String	Read/Write
Id	String	Read/Write

Event package string is used for packages that are not listed in the Enum Event Package table (see Event Package on page 175).

Host

The host structure is applicable to the URL structure (see 'URL' on page URL on page 173) and the Via header (see 'Via' on page Via on page 168).

Table 6-3: Host Structure

Keyword	Sub Types
Port	Short
Name	String

MLPP

This structure is applicable to the Reason header (see Reason on page 153).

Table 6-4: MLPP Structure

Keyword	Sub Types
Type	Enum MLPP Reason (see MLPP Reason Type on page 176)

Keyword	Sub Types
Cause	Int

Privacy Struct

This structure is applicable to the Privacy header (see Privacy on page 151).

Table 6-5: Privacy Structure

Keyword	Sub Types
NONE	Boolean
HEADER	Boolean
SESSION	Boolean
USER	Boolean
CRITICAL	Boolean
IDENTITY	Boolean
HISTORY	Boolean

Reason Structure

This structure is applicable to the Reason header (see Reason on page 153).

Table 6-6: Reason Structure

Keyword	Sub Types
Reason	Enum Reason (see Reason (Reason Structure) on page 178)
Cause	Int
Text	String

SIPCapabilities

This structure is applicable to the following headers:

- Supported (see Supported on page 165)
- Require (see Require on page 159)
- Proxy-Require (see Proxy-Require on page 152)
- Unsupported (see Unsupported on page 167)

Table 6-7: SIPCapabilities Structure

Keyword	Sub Types
EarlyMedia	Boolean
ReliableResponse	Boolean
Timer	Boolean
EarlySession	Boolean
Privacy	Boolean
Replaces	Boolean
History	Boolean
Unknown	Boolean
GRUU	Boolean
ResourcePriority	Boolean
TargetDialog	Boolean
SdpAnat	Boolean

URL

This structure is applicable to the following headers:

- Contact (see Contact on page 140)
- Diversion (see Diversion on page 141)
- From (see From on page 143)
- P-Asserted-Identity (see P-Asserted-Identity on page 147)
- P-Associated-Uri (see P-Associated-Uri on page 148)
- P-Called-Party-Id (see P-Called-Party-Id on page 148)
- P-Preferred-Identity (see P-Preferred-Identity on page 150)
- Referred-By (see Referred-By on page 154)
- Refer-To (see Refer-To on page 155)
- Remote-Party-Id (see Remote-Party-Id on page 156)
- Request-Uri (see Request-Uri on page 157)
- To (see To on page 166)

Table 6-8: URL Structure

Keyword	Sub Types
Type	Enum Type (see Type on page 182)
Host	Host Structure (see Host on page 171)
MHost	Structure
UserPhone	Boolean
LooseRoute	Boolean
User	String
TransportType	Enum Transport (see TransportType on page 182)
Param	Param

Random Type

Manipulation rules can include random strings and integers. An example of a manipulation rule using random values is shown below:

```
MessageManipulations 4 = 1, Invite.Request, , Header.john, 0, rand.string.56.A.Z, 0;
```

In this example, a header called "john" is added to all INVITE messages received by the device and a random string of 56 characters containing characters A through Z is added to the header.

For a description of using random values, see the subsequent subsections.

Random Strings

The device can generate random strings in header manipulation rules that may be substituted where the type 'String' is required. The random string can include up to 298 characters and include a range of, for example, from a to z or 1 to 10. This string is used in the table's 'Action Value' field.

The syntax for using random strings is:

```
Rand.string.<number of characters in string>.<low character>.<high character>
```

Examples:

- Rand.string.5.a.z: This generates a 5-character string using characters a through z.
- Rand.string.8.0.z: This generates an 8-character string using characters and digits.

Random Integers

The device can generate a random numeric value that may be substituted where the type 'Int' is required. The syntax for random numeric values is:

```
Rand.number.<low number>.<high number>
```

Examples:

Rand.number.5.32: This generates an integer between 5 and 32

Enum Definitions

AgentRole

These ENUMs are applicable to the Server or User-Agent headers (see Server or User-Agent on page 161).

Table 6-9: Enum Agent Role

AgentRole	Value
Client	1
Server	2

Event Package

These ENUMs are applicable to the Server or User-Agent (see Server or User-Agent on page 161) and Event (see Event on page 143) headers.

Table 6-10: Enum Event Package

Package	Value
TELEPHONY	1
REFER	2
REFRESH	3
LINE_STATUS	4
MESSAGE_SUMMARY	5
RTCPXR	6
SOFT_SYNC	7

Package	Value
CHECK_SYNC	8
PSTN	9
DIALOG_PACKAGE	10
REGISTRATION	11
START_CWT	12
STOP_CWT	13
UA_PROFILE	14
LINE_SEIZE	15

MLPP Reason Type

These ENUMs are applicable to the MLPP Structure (see MLPP on page 171).

Table 6-11: Enum MLPP Reason Type

Type	Value
PreEmption Reason	0
MLPP Reason	1

Number Plan

These ENUMs are applicable to the Remote-Party-Id header (see Remote-Party-Id on page 156).

Table 6-12: Enum Number Plan

Plan	Value
ISDN	1
Data	3
Telex	4
National	8
Private	9
Reserved	15

Number Type

These ENUMs are applicable to the Remote-Party-Id header (see Remote-Party-Id on page 156).

Table 6-13: Enum Number Type

Number Type	Value
INTERNATIONAL LEVEL2 REGIONAL	1
NATIONAL LEVEL1 REGIONAL	2
NETWORK PISN SPECIFIC NUMBER	3
SUBSCRIBE LOCAL	4
ABBREVIATED	6
RESERVED EXTENSION	7

Privacy

These ENUMs are applicable to the Remote-Party-Id (see Remote-Party-Id on page 156) and Diversion (see Diversion on page 141) headers.

Table 6-14: Enum Privacy

Privacy Role	Value
Full	1
Off	2

Reason (Diversion)

These ENUMs are applicable to the Diversion header (see Diversion on page 141).

Table 6-15: Enum Reason

Reason	Value
Busy	1
No Answer	2
Unconditional	3
Deflection	4
Unavailable	5

Reason	Value
No Reason	6
Out of service	7

Reason (Reason Structure)

These ENUMs are used in the Reason Structure (see Reason Structure on page 172).

Table 6-16: Enum Reason (Reason Structure)

Reason	Value
INVITE	5
REINVITE	6
BYE	7
OPTIONS	8
ACK	9
CANCEL	10
REGISTER	11
INFO	12
MESSAGE	13
NOTIFY	14
REFER	15
SUBSCRIBE	16
PRACK	17
UPDATE	18
PUBLISH	19
LAST_REQUEST	20
TRYING_100	100
RINGING_180	180

Reason	Value
CALL_FORWARD_181	181
QUEUED_182	182
SESSION_PROGRESS_183	183
OK_200	200
ACCEPTED_202	202
MULTIPLE_CHOICE_300	300
MOVED_PERMANENTLY_301	301
MOVED_TEMPORARILY_302	302
SEE_OTHER_303	303
USE_PROXY_305	305
ALTERNATIVE_SERVICE_380	380
BAD_REQUEST_400	400
UNAUTHORIZED_401	401
PAYMENT_REQUIRED_402	402
FORBIDDEN_403	403
NOT_FOUND_404	404
METHOD_NOT_ALLOWED_405	405
NOT_ACCEPTABLE_406	406
AUTHENTICATION_REQUIRED_407	407
REQUEST_TIMEOUT_408	408
CONFLICT_409	409
GONE_410	410
LENGTH_REQUIRED_411	411
CONDITIONAL_REQUEST_FAILED_412	412

Reason	Value
REQUEST_TOO_LARGE_413	413
REQUEST_URI_TOO_LONG_414	414
UNSUPPORTED_MEDIA_415	415
UNSUPPORTED_URI_SCHEME_416	416
UNKNOWN_RESOURCE_PRIORITY_417	417
BAD_EXTENSION_420	420
EXTENSION_REQUIRED_421	421
SESSION_INTERVAL_TOO_SMALL_422	422
SESSION_INTERVAL_TOO_SMALL_423	423
ANONYMITY_DISALLOWED_433	433
UNAVAILABLE_480	480
TRANSACTION_NOT_EXIST_481	481
LOOP_DETECTED_482	482
TOO_MANY_HOPS_483	483
ADDRESS_INCOMPLETE_484	484
AMBIGUOUS_485	485
BUSY_486	486
REQUEST_TERMINATED_487	
NOT_ACCEPTABLE_HERE_488	488
BAD_EVENT_489	489
REQUEST_PENDING_491	491
UNDECIPHERABLE_493	493
SECURITY_AGREEMENT_NEEDED_494	494
SERVER_INTERNAL_ERROR_500	500

Reason	Value
NOT_IMPLEMENTED_501	501
BAD_GATEWAY_502	502
SERVICE_UNAVAILABLE_503	503
SERVER_TIME_OUT_504	504
VERSION_NOT_SUPPORTED_505	505
MESSAGE_TOO_LARGE_513	513
PRECONDITION_FAILURE_580	580
BUSY_EVERYWHERE_600	600
DECLINE_603	603
DOES_NOT_EXIST_ANYWHERE_604	604
NOT_ACCEPTABLE_606	606

Reason (Remote-Party-Id)

These ENUMs are applicable to the Remote-Party-Id header (see Remote-Party-Id on page 156).

Table 6-17: Enum Reason (RPI)

Reason	Value
Busy	1
Immediate	2
No Answer	3

Refresher

These ENUMs are used in the Session-Expires header (see Session-Expires on page 163).

Table 6-18: Enum Refresher

Refresher String	Value
UAC	1
UAS	2

Screen

These ENUMs are applicable to the Remote-Party-Id (see Remote-Party-Id on page 156) and Diversion (see Diversion on page 141) headers.

Table 6-19: Enum Screen

Screen	Value
Yes	1
No	2

ScreenInd

These ENUMs are applicable to the Remote-Party-Id header (see Remote-Party-Id on page 156).

Table 6-20: Enum ScreenInd

Screen	Value
User Provided	0
User Passed	1
User Failed	2
Network Provided	3

TransportType

These ENUMs are applicable to the URL Structure (see URL on page 173) and the Via header (see Via on page 168).

Table 6-21: Enum TransportType

TransportType	Value
UDP	0
TCP	1
TLS	2
SCTP	3

Type

These ENUMs are applicable to the URL Structure (see URL on page 173).

Table 6-22: Enum Type

Type	Value
SIP	1
Tel	2
Fax	3
SIPS	4

Address Presentation Restricted Indicator

These ENUMs are applicable to the phone number handling (see ISUP Body Manipulation).

Table 6-23: Enum Presentation Restricted Indicator

Presentation	Value
Allowed	0
Restricted	1

Transmission Medium Requirement

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-24: Enum Transmission Medium Requirement

Transmission Medium Requirement (TMR)	Value
Speech	0
64 kbits/s unrestricted	2
3.1 kHz audio	3

Charge Indicator

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-25: Enum Charge Indicator

Charge Indicator	Value
No indication	0
No charge	1

Charge Indicator	Value
Charge	2

Called Party Status Indicator

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-26: Enum Called Party Status Indicator

Called Party Status Indicator	Value
No indication	0
Subscriber free	1

Called Party Category Indicator

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-27: Enum Called Party Category Indicator

Called Party Category Indicator	Value
Ordinary subscriber	0
Test call	40
Priority	41
Payphone	70
No indication	71

Event Information

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-28: Enum Event Information

Event Information	Value
No INFORMATION	0
ALERTING	1
PROGRESS	2
In-band information	3

Cause Value

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-29: Enum Cause Value

Cause	Value
Unallocated number	1
No route to specified transit network	2
No route to destination	3
Send special information tone	4
Misdialled trunk prefix	5
Channel unacceptable	6
Call awarded and being delivered in an established channel	7
Preemption	8
Preemption – circuit reserved for reuse	9
Normal call clearing	16
User busy	17
No user responding	18
No answer from user (user alerted)	19
Subscriber absent	20
Call rejected	21
Number changed	22
Redirection to new destination	23
Exchange routing error	25
Non-selected user clearing	26
Destination out of order	27
Invalid number format (address incomplete)	28

Cause	Value
Facility rejected	29
Response to STATUS ENQUIRY	30
Normal, unspecified	31
No circuit/channel available	34
Network out of order	38
Permanent frame mode connection out of service	39
Permanent frame mode connection operational	40
Temporary failure	41
Switching equipment congestion	42
Access information discarded	43
Requested circuit/channel not available	44
Precedence call blocked	46
Resource unavailable, unspecified	47
Quality of service not available	49
Requested facility not subscribed	50
Outgoing calls barred within CUG	53
Incoming calls barred within CUG	55
Bearer capability not authorized	57
Bearer capability not presently available	58
Inconsistency in designated outgoing access information and subscriber class	62
Service or option not available, unspecified	63
Bearer capability not implemented	65
Channel type not implemented	66
Requested facility not implemented	69

Cause	Value
Only restricted digital information bearer capability is available	70
Service or option not implemented, unspecified	79
Invalid call reference value	81
Identified channel does not exist	82
A suspended call exists, but this call identity does not	83
Call identity in use	84
No call suspended	85
Call having the requested call identity has been cleared	86
User not member of CUG	87
Incompatible destination	88
Non-existent CUG	90
Invalid transit network selection	91
Invalid message, unspecified	95
Mandatory information element is missing	96
Message type non-existent or not implemented	97
Message not compatible with call state or message type non-existent or not implemented	98
Information element /parameter nonexistent or not implemented	99
Invalid information element contents	100
Message not compatible with call state	101
Recovery on timer expiry	102
Parameter non-existent or not implemented, passed on	103
Message with unrecognized parameter, discarded	110
Protocol error, unspecified	111
Interworking, unspecified	127

Cause Location

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation).

Table 6-30: Enum Cause Location

Location	Value
user (U)	0
private network serving the local user (LPN)	1
public network serving the local user (LN)	2
transit network (TN)	3
public network serving the remote user (RLN)	4
private network serving the remote user (RPN)	5
international network (INTL)	7
network beyond interworking point (BI)	10

Redirect Reason

These ENUMs are applicable to the ISUP handling (see ISUP Body Manipulation on page 83).

Table 6-31: Enum Redirect Reason

Redirect Reason	Value
Busy	1
No reply	2
Deflection	4
Deflection Immediate	5
Mobile subscriber not reachable	6
Unconditional	15

Actions and Types

Table 6-32: Action and Types

Element Type	Command Type	Command	Value Type	Remarks
IPGroup	Match	==	String	Returns true if the parameter equals to the value.
		!=	String	Returns true if the parameter not equals to the value.
		contains	String	Returns true if the string given is found in the parameter value.
		!contains	String	Returns true if the string given is not found in the parameter value.
Call-Parameter	Match	==	String	Returns true if the parameter equals to the value.
		!=	String	Returns true if the parameter not equals to the value.
		contains	String	Returns true if the string given is found in the parameter value.
		!contains	String	Returns true if the string given is not found in the parameter value.
Body	Match	==	String	Returns true if the body's content equals to the value.
		!=	String	Returns true if the body's content not equals to the value.
		contains	String	Returns true if the

Element Type	Command Type	Command	Value Type	Remarks
				string given is found in the body's content.
		!contains	String	Returns true if the string given is not found in the body's content.
		exists		Returns true if this body type exists in the message.
		!exists		Returns true if this body type does not exist in the message.
	Action	Modify	String	Modifies the body content to the new value.
		Add	String	Adds a new body to the message. If such body exists the body content will be modified.
		Remove		Removes the body type from the message.
Header-List	Match	==	String *Header-list	Returns true if the header's list equals to the string.
		!=	String *Header-list	Returns true if the header's list not equals to the string.
		contains	String	Returns true if the header's list contains the string.
		!contains	String	Returns true if the header's list does not contain the string.

Element Type	Command Type	Command	Value Type	Remarks
		exists		Returns true if at least one header exists in the list.
		!exists		Returns true if no headers exist in the list.
	Action	Modify	String *Header	Removes all the headers from the list and allocates a new header with the given value.
		Add	String *Header	Adds a new header to the end of the list.
		Remove		Removes the whole list from the message.
Header	Match	==	String *Header	Returns true if a header equals to the value. The header element must not be a list.
		!=	String *Header	Returns true if a header not equals to the value. The header element must not be a list.
		contains	String	Returns true if the header contains the string.
		!contains	String	Returns true if the header does not contain the string.
		exists		Returns true if the header exists.
		!exists		Returns true if the header does not exist.
	Action	Modify	String	Replaces the entire

Element Type	Command Type	Command	Value Type	Remarks
			*Header	header with the new value.
		Remove		Removes the header from the message, if the header is part of a list only that header will be removed.
		Add	String *Header	Adds a new header to the end of the list.
Parameter-List	Match	==	String Parameter-list	Returns true if the header's list equals to the string.
		!=	String Parameter-list	Returns true if the header's list not equals to the string.
		contains	String	Returns true if the header's list contains the string.
		!contains	String	Returns true if the header's list does not contain the string.
		exists		Returns true if at least one parameter exists in the list.
		!exists		Returns true if the header's parameter list is empty.
	Action	Modify	String Parameter-list	Replaces the current parameters with the new value.
		Add	String Parameter	Adds a new parameter to the parameter's list.

Element Type	Command Type	Command	Value Type	Remarks
		Remove		Removes all the unknown parameters from the list.
Parameter	Match	==	String Parameter	Returns true if the header's parameter's value equals to the value.
		!=	String Parameter	Returns true if the header's parameter's value not equals to the value.
		contains	String	Returns true if the header's parameter contains the string.
		!contains	String	Returns true if the header's parameter does not contain the string.
		exists		Returns true if the header's parameter exists.
		!exists		Returns true if the header's parameter does not exist.
	Action	Modify	String Parameter	Sets the header's parameter to the value.
		Remove		Removes the header's parameter from the parameter list.
Structure	Match	==	String *Structure	Returns true if the header's structure's value equals to the value. The string given must

Element Type	Command Type	Command	Value Type	Remarks
				be able to be parsed to the structure.
		!=	String *Structure	Returns true if the header's structure's value not equals to the value. The string given must be able to be parsed to the structure.
	Action	Modify	String *Structure	Sets the header's structure to the value. The string given must be able to be parsed to the structure.
Integer	Match	==	Integer	Returns true if value equals to the integer element
		!=	Integer	Returns true if value not equals to the integer element
		>	Integer	Returns true if value is greater than the value.
		>=	Integer	Returns true if value is greater than or equals to the value.
		<	Integer	Returns true if value is less than the value.
		<=	Integer	Returns true if value is less than or equals to the value.
	Action	Modify	Integer	Sets the integer element to the value. A string value must be a representation of an

Element Type	Command Type	Command	Value Type	Remarks
				integer.
String	Match	==	String	Returns true if the string element equals to the value.
		!=	String	Returns true if the string element not equals to the value.
		contains	String	Returns true if the value is found in the string element.
		!contains	String	Returns true if the value is not found in the string element.
		>	String	Performs a character by character compare. Returns true if the ASCII value of the character is greater than that in the value
		>=	String	Performs a character by character compare. Returns true if the ASCII value of the character is greater than or equal to that in the value
		<	String	Performs a character by character compare. Returns true if the ASCII value of the character is less than that in the value
		<=	String	Performs a character by character compare. Returns true if the ASCII value of the character is

Element Type	Command Type	Command	Value Type	Remarks
				less than or equal to that in the value
	Action	Modify	String	Sets the string element to the value.
		Add prefix	String	Adds the value to the beginning of the string element.
		Remove prefix	String	Removes the value from the beginning of the string element.
		Add suffix	String	Adds the value to the end of the string element.
		Remove suffix	String	Removes the value from the end of the string element.
Boolean	Match	==	Boolean	Returns true if the Boolean element equals to the value. Boolean – can be either 0 or 1.
		!=	Boolean	Returns true if the Boolean element not equals to the value. Boolean – can be either 0 or 1.
		>	Boolean	Returns true if the Boolean element not equals to the value. Boolean – can be either 0 or 1.
		<	Boolean	Returns true if the Boolean element not equals to the value.

Element Type	Command Type	Command	Value Type	Remarks
				Boolean – can be either 0 or 1.
	Action	Modify	Boolean	Sets the Boolean element to the value. Boolean – can be either 0 or 1.
Attribute	Match	==	Integer *Attribute	Returns true if the attribute element equals to the value. An attribute element value must be of the same type of the attribute element.
		!=	Integer *Attribute	Returns true if the attribute element not equals to the value. An attribute element value must be of the same type of the attribute element.
	Action	Modify	Integer *Attribute	Sets the attribute element to the value. An attribute element value must be of the same type of the attribute element.

Syntax

This section describes the fields of the message manipulation tables:

Message Type

Description: Rule is applied only if this is the message's type

Syntax: <method>.<message role>

- Method:
  - Description: Rule is applied only if this is the message's method

- Syntax: token / any
- Examples:
  - invite: rule applies only to INVITE messages
  - MyProprietary: rule applies only to the non-standard MyProprietary message
  - any: no limitation on the method type

- Message role:
  - Description: Rule is applied only if this is the message's role
  - Syntax: request / response.response-code / any
  - Examples:
    - request: rule applies only on requests
    - response.200: rule applies only on 200 OK messages
    - any: no limitations on the type of the message
- Response code:
  - Description: Response code of the message
  - Syntax: 1xx / 18x / 2xx / 3xx / 4xx / 5xx / 6xx / 3digit / any
  - Examples:
    - 3xx: any redirection response
    - 18x: any 18x response
    - 200: only 200 OK response
    - Any: any response

Examples:

- invite.request
- invite.response.200
- invite.response.18x
- subscribe.response.2xx

Condition

Description: Matching criteria for the rule

Syntax: (Action Subject / param) SWS match-type [SWS Action Value] * [SWS logical-expression SWS Condition]

Examples:

- header.from.url.user == '100'
- header.contact.header-param.expires > '3600'

- header.to.host contains 'itsp'
- param.call.dst.user != '100'
- header.john exists
- header.john exists AND header.to.host !contains 'john'
- header.from.url.user == '100' OR header.from.url.user == '102' OR header.from.url.user == '300'
- match-type
  - Description: Comparison to be made
  - Syntax:
    - ==: equals
    - !=: not equals
    - >: greater than
    - <: less than
    - >=: greater than or equal to
    - <=: less than or equal to
    - contains: does a string contain a value (relevant only to string fields)
    - exists: does a certain header exists
    - !exists: does a certain header not exists
    - !contains: does a string exclude a value. Relevant only to string fields

When comparing SIP header elements that are strings, the operators '>', '<', and '=' can evaluate the expression using either lexicographical (alphanumeric) comparison or numerical value comparison. For numerical comparion, you need to use the 'num' keyword (num>, num<, num=, num>=, and so on).
For example, if the To header in a SIP message is:

```
To: sip:100@10.33.40.88;user=phone
```

and you have two Message Manipulation rules with the following conditions:

- **Rule 1**: Header.to.url.user > '20'
- **Rule 2:** Header.to.url.user num> '20'

Then:

- Rule 1 doesn't match because '100' is not lexicographically greater than ‘20’ (i.e., 1 is less than 2).
- Rule 2 matches because ’100’ is numerically greater than ‘20’.

- logical-expression:
  - Description: Condition for the logical expression
  - Syntax:
    - AND: logical operator AND

- OR: logical operator OR

- Expressions are evaluated from left to right. For example: "A AND B OR C" is calculated as (A AND B) OR C.
- The use of AND and OR is limited to four appearances in a single condition. For example: "A or B and C or D or E" is valid; "A or B and C or D and E or F" is invalid.

Action Subject

Description: Element in the message

Syntax: (header / body).Action Subject name [.header-index] * [.(sub-element / sub-element-param)]

Examples:

- header.from
- header.via.2.host
- header.contact.header-param.expires
- header.to.uri-param.user-param
- body.application/dtmf-relay
- Action Subject name:
  - Description: Name of the message's element - "/" only used for body types
  - Syntax: 1 * (token / "/")
  - Examples:
    - from (header's name)
    - to (header's name)
    - application/dtmf-relay (body's name)
- header-index:
  - Description: Header's index in the list of headers
  - Syntax: Integer
  - Examples: If five Via headers arrive:
    - 0 (default) refers to first Via header in message
    - 1: second Via header
    - 4: fifth Via header
- sub-element:
  - Description: Header's element

- Syntax: sub-element-name
- Examples:
  - user
  - host

- sub-element-param:
  - Description: Header's element
  - Syntax: sub-element-name [.sub-element-param-name]
  - Example:
    - header.from.param.expires
- sub-element-param-name
  - Description: Header's parameter name - relevant only to parameter sub-elements
  - Syntax: token
  - Examples:
    - expires (contact's header's param)
    - duration (retry-after header's param)
    - unknown-param (any unknown param can be added/removed from the header)
- param:
  - Description: Params can be as values for match and action
  - Syntax: param.param-sub-element.param-dir-element.(call-param-entity / ipg-param-entity)
  - Examples:
    - param.ipg. src.user
    - param.ipg.dst.host
    - param.ipg.src.type
    - param.call.src.user
- param-sub-element:
  - Description: Determines whether the param being accessed is a call or an IP Group
  - Syntax:
    - call: relates to source or destination URI for the call
    - ipg: relates to source or destination IP Group
- param-dir-element:
  - Description: Direction relating to the classification

- Syntax:
  - src: refers to source
  - ds: refers to destination

- call-param-entity
  - Description: Parameters that can be accessed on the call
  - Syntax:
    - user: refers to username in request-URI for call
- ipg-param-entity:
  - Description: Name of the parameter
  - Syntax:
    - user: refers to Contact user in IP Group
    - host: refers to Group Name in IP Group table
    - type: refers to Type field in IP Group table
    - id: refers to IP Group ID (used to identify source or destination IP Group)
- string:
  - Description: String
  - Syntax: string enclosed in single apostrophe
  - Examples:
    - 'username
    - '123'
    - 'user@host'
- Integer:
  - Description: A number
  - Syntax: 1 * digit
  - Example:
    - 123

Action Type

Description: Action to be performed on the element

Syntax:

- modify: sets element to new value (all element types)
- add-prefix: adds value at beginning of string (string element only)

- remove-prefix: removes value from beginning of string (string element only)
- add-suffix: adds value at end of string (string element only)
- remove-suffix: removes value from end of string (string element only)
- add: adds a new header/param/body (header or parameter elements)
- remove: removes a header/param/body (header or parameter elements)

Action Value

Description: Value for action and match

Syntax: ('string' / Action Subject / param) * (+ ('string' / Action Subject / param))

Examples:

- 'itsp.com'
- header.from.url.user
- param.ipg.src.user
- param.ipg.dst.host + '.com'
- param.call.src.user + '<' + header.from.url.user + '@' + header.p-asserted-identity.url.host + '>'

This page is intentionally left blank.

International Headquarters

6 Ofra Haza Street

Naimi Park

Or Yehuda, 6032303, Israel

Tel: +972-3-976-4000

Fax: +972-3-976-4040

AudioCodes Inc.

80 Kingsbridge Rd

Piscataway, NJ 08854, USA

Tel: +1-732-469-0880

Fax: +1-732-469-2298

Contact us: https://www.audiocodes.com/corporate/offices-worldwide

Website: https://www.audiocodes.com/

Documentation Feedback: https://online.audiocodes.com/documentation-feedback

Document #: LTRT-29073