Session Border Controllers

Multi-Service Business Routers

VoIP Media Gateways

Quick Reference Guide

SIP Message Manipulation

Version 6.6

August 2013

Document # LTRT-28623

AudioCodes

Table of Contents

List of Tables

Notice

This document describes the Quick Reference Guide for configuring SIP Message Manipulation.

Information contained in this document is believed to be accurate and reliable at the time of printing. However, due to ongoing product improvements and revisions, AudioCodes cannot guarantee accuracy of printed material after the Date Published nor can it accept responsibility for errors or omissions. Before consulting this document, check the corresponding Release Notes regarding feature preconditions and/or specific support in this release. In cases where there are discrepancies between this document and the Release Notes, the information in the Release Notes supersedes that in this document. Updates to this document and other documents as well as software files can be downloaded by registered customers at http://www.audiocodes.com/downloads.

Date Published: August-07-2013

Trademarks

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.

Customer Support

Customer technical support and service are generally provided by AudioCodes' Distributors, Partners, and Resellers from whom the product was purchased. For technical support for products purchased directly from AudioCodes, or for customers subscribed to AudioCodes Customer Technical Support (ACTS), contact support@audiocodes.com.

Abbreviations and Terminology

Each abbreviation, unless widely used, is spelled out in full when first used.

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 Web site at http://www.audiocodes.com/downloads. Your valuable feedback is highly appreciated.

Related Documentation

Manual Name
MediaPack User's Manual
Mediant 600 and 1000 SIP User's Manual
Mediant 800 Gateway and E-SBC User's Manual
Mediant 800 MSBR User's Manual
Mediant 850 MSBR User's Manual
Mediant 1000B Gateway and E-SBC User's Manual
Mediant 1000B MSBR User's Manual
Mediant 2000 User's Manual
Mediant 3000 User's Manual
Mediant 4000 E-SBC User's Manual
Mediant Software E-SBC User's Manual

1 Introduction

This document provides a quick reference on how to configure SIP Message Manipulations using AudioCodes embedded Web server. The document includes examples that have been implemented in the field.

The following topics are covered in this document:

- Message Manipulation table fields - see Section 2 on page 12.
- Detailed Field Syntax - see Section 3 on page 15.
- Typical Examples - see Section 4 on page 29.

For a detailed configuration of SIP Message Manipulations, refer to the relevant product's *User's Manual*.

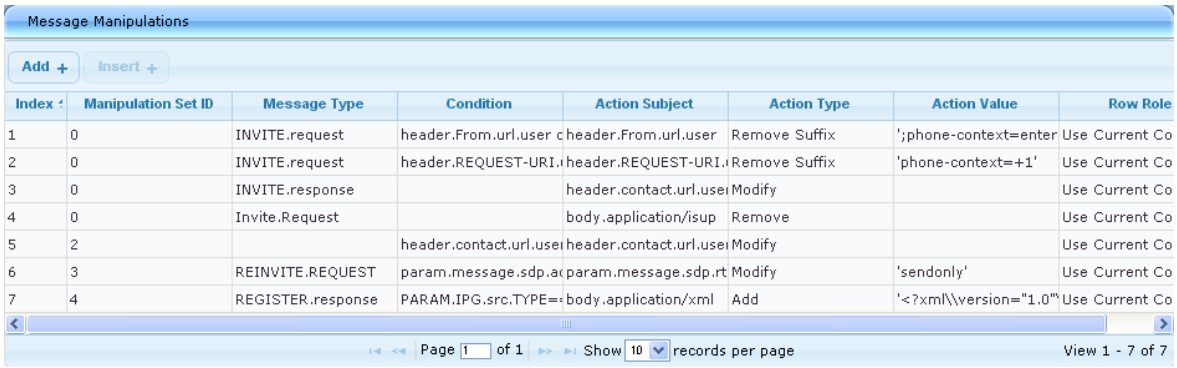
Reader's Notes

2 Message Manipulation Table Fields

SIP Message Manipulation is configured in the Message Manipulation table in the AudioCodes embedded Web server (**Configuration** tab > **VoIP** > **SIP Definitions** > **Msg Policy & Manipulation** > **Message Manipulations**).

The figure below shows an example of SIP Message Manipulation rules in this table.

Figure 2-1: Message Manipulation Table



This section describes the Message Manipulation table fields and their syntax used for entering the values:

- Manipulation Set ID – see Section 2.1 on page 10
- Message Type – see Section 2.2 on page 11
- Condition – see Section 2.3 on page 12
- Action - See Section 2.4 on page 13
  - Action Subject
  - Action Type
  - Action Value

2.1 Manipulation Set ID

The 'Manipulation Set ID' field enables you to group message manipulation rules that you have defined. Once you have defined manipulation rules and associated them with a specific Manipulation Set ID, you **must** assign this ID to the relevant IP Group in the IP Group table, where they can be assigned to either the inbound (Inbound Message Manipulation Set) or outbound (Outbound Message Manipulation Set) leg.

Syntax:

```
<0-19>
```

where:

- <0-19> specifies the Manipulation Set ID. You can define up to 20 message manipulation rule sets and up to 100 rules (there is no rule limit per set).

2.2 Message Type

The following syntax determines the type of message to which the manipulation rule refers.

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.

The following table provides examples of different message types.

Table 2-1: Message Types Examples and Descriptions

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

2.3 Condition

The 'Condition' field is used to test specific parts of the header in the message with specified values. Conditions may be combined with other conditions using logical operators (and/or).

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

The following table provides various examples of different conditions.

Table 2-2: Condition Examples and Descriptions

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.address == '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>.

2.4 Action

The following describes the syntax of the 'Action' field:

Syntax:

```
<Action Subject>
```

where:

- **<Action Subject>** specifies the message component upon which you wish to manipulate, using the following format:

  header/body/variable

Syntax:

```
<Action Type>
```

where:

- **<Action Type>** specifies the type of action you wish to perform on the message component, using the following format:

  action-operand

Syntax:

```
<Action Value>
```

where:

- **<Action Value>** specifies the value to assign to the Action Type and Action Subject, using the following format:

  string/header/body/parameter/random/variable/regex

The following table provides various example actions.

Table 2-3: Action Examples and Descriptions

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	Delete		Deletes the header "customername" from the message.
var.global.0	Modify	header.user-agent.content	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 Contact header and assigns the value "Audiocodes" to it.

Reader's Notes

3 Detailed Field Syntax

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

- **Condition Operands** – see Section 3.1 below.
- **Action Operands** – see Section 3.2 below.
- **Strings** – see Section 3.3 on page 16.
- **Headers** – see Section 3.4 on page 17.
- **Body** – see Section 3.5 on page 20.
- **Parameters** – see Section 3.6 on page 22.
- **Variables** – see Section 3.7 on page 25.
- **Random Characters** – See Section 3.8 on page 26.
- **Regular Expressions** – See Section 3.9 on page 27.

3.1 Condition Operands

The following table describes the condition operands.

Table 3-1: Condition Operands and Descriptions

Condition Operand	Description
== / !=	Tests for equivalent / not equivalent values.
>= / <=	Tests for greater than or equal to / less than or equal to values.
> / <	Tests for greater than / less than values.
contains / !contains	Tests a string containing / not containing specified text.
exists /!exists	Tests whether a parameter exists / does not exist.
Suffix / prefix	Tests whether a string has a particular suffix / prefix.
len> / len< / len==	Tests whether the length of a string is greater than / less than / equal to a specific value.
regex	Tests whether a string matches the given regular expression.

3.2 Action Operands

The following table describes the action operands.

Table 3-2: Action Operands and Descriptions

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.

3.3 Strings

The string type is the most basic of all syntax types. A string is a series of characters enclosed by single apostrophe. It can be used as the value for the following Message Manipulation table fields:

- Condition
- Action Value

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

Table 3-3: Configuration Examples of Using Strings in Message Manipulations Table

Message Type	Condition	Action Subject	Action Type	Action Value
invite.request	header.user-agent.content contains 'X-Lite'	header.user-agent.content	Modify	'anonymous UA'
invite.request	header.from.url.user=='101;ext=7166'	header.user-agent.content	Modify	'anonymous UA'

3.4 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. 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.

  For a complete list of all the sub-types available for each header, refer to the "Message Manipulation" Section in the relevant *User's Manual.*

3.4.1 Header Field Examples

The following table provides examples of header fields.

Table 3-4: Header Fields Examples and Descriptions

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 a To header.
header.from.name	Defines the display name in the From header.
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 1st Remote-Party-ID header.
header.contact.3	Defines the 3rd Contact header.
header.via.2.url.user	Defines the user part of the 2nd Via header.

3.4.2 Configuration Examples

The following table provides configuration examples for using header fields in the Message Manipulations table.

Table 3-5: Configuration Examples for using Header Fields in Message Manipulations Table

Message Type	Condition	Action Subject	Action Type	Action Value
register. request	header.from.url. user == '101' OR header.from.url. user == '1000'	header.from.url.user	Modify	'2000'
register		header.to.url.host. name	Modify	'audiocodes.com'
invite		header.from.name	Modify	header.contact. url.user
invite. request		header.newheader	Add	'information to client'
subscribe	header.via.tran sporttype=='1'	header.to.param .transporttype	Add	'TCP'

3.5 Body

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

Syntax:

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

where:

<body-name> specified the body name as it arrives in the message. For example, 'application/sdp' (case-insensitive).

3.5.1 Body Examples

The following table provides examples of the message body.

Table 3-6: Message Body Examples and Descriptions

Subject	Description
body.application/ x-nt-mcdn-frag-hex	Adds or removes this 'unknown' body type.
body.sdp	Defines the SDP in the body.

3.5.2 Configuration Examples

The following table provides configuration examples for the message body in the Message Manipulations table.

Table 3-7: Configuration Examples for Message Body in the Message Manipulations Table

Message Type	Condition	Action Subject	Action Type	Action Value
invite	body.sdp !exists	body.application/ x-nt-mcdn-frag-hex	Add	'a=0981233\\b=12rew wer\\note=newlinecharacter'
invite.request		Body.mwi	Add	'Messages-Waiting: yes\\Message-Account: sip:alice@vmail.example.com\\Voice-Message: 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'

Message Type	Condition	Action Subject	Action Type	Action Value
any		body.mwi.summary.oldurgentmsgs	Modify	'67'
invite		body.mwi.pending	Modify	'8'
invite		body.mwi.message waiting	Modify	'2'

3.6 Parameters

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

- Message Parameters
- IP Group Parameters
- Call Parameters

3.6.1 Message Parameter Syntax

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

Table 3-8: Message Parameter Syntax in the Message Manipulations Table

Subject	Description
param.message.sdp.address	Specifies the address in the SDP.
param.message.sdp.rtpmode	Specifies the RTP mode in the SDP.
param.message.sdp.originaddress	Specifies the origin address in the SDP.
param.message.sdp.port	Specifies the port in the SDP.
param.message.address.<src/dst>.port	Specifies the port as a string for the source or destination of the message.
param.message.address.<src/dst>.address	Specifies the IP address as a string for the source or destination of the message.
param.message.address.<src/dst>.<transporttype>	Specifies the transport type as a string for the source or destination of the message. where **<transporttype>** is one of the following values: ▪ UDP ▪ TCP ▪ TLS

3.6.2 IP Group Parameter Syntax

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

Table 3-9: IP Group Parameter Syntax in Message Manipulations Table

Subject	Description
param.ipg.<src/dst>.user	Specifies the source or destination contact address for an active call.
param.ipg.<src/dst>.host	Specifies the source or destination group name for an active call.
param.ipg.<src/dst>.type	Specifies the source or destination group type an active call. where <src/dst> is one of the following values: ▪ Server ▪ User ▪ gateway
param.ipg.<src/dst>.id	Specifies the source or destination group number as a string for an active call.

3.6.3 Call Parameter Syntax

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

Table 3-10: Call Parameter Syntax in Message Manipulations Table

Subject	Description
param.call.<src/dst>.user	Specifies the source or destination username during run-time.

3.6.4 Configuration Examples

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

Table 3-11: Configuration Examples using Parameters in Message Manipulations Table

Message Type	Condition	Action Subject	Action Type	Action Value
	param.message.sdp.address == '10.132.10.101'	header.IPSource	Add	param.ipg.src.id
invite.response.200	param.message.sdp.rtpmode=='inactive'	header.origin	Add	param.message.sdp.originaddress
	param.message.sdp.rtpmode== 'inactive'	header.from.param.origin	Add	param.message.sdp.originaddress
subscribe.request		header.to.param.user	Add	param.call.src.user
invite.response		header.request-uri.url.param.myname	Add	param.ipg.src.host

3.7 Variables

There are two types of variables used in the Message Manipulation tables:

- **Call** variables are used to store information throughout the lifetime of a call; SRC or DST references which can be stored in the call leg. Note data stored in the call variables is only valid for the duration of the call.
- **Global** variables, which are similar to call variables; however, their lifetime is not restricted to the duration of a call.

The following syntax shows how to specify the call source variable.

Syntax:

```
var.call.src.<0>
```

where:

<0> specifies the variable ID (note that only one source call variable can be defined).

The following syntax shows how to specify the call destination variable.

Syntax:

```
var.call.dst.<0>
```

where:

<0> specifies the variable ID (note that only one destination call variable can be defined).

The following syntax shows how to specify the global variables.

Syntax:

```
var.global.<0-9>
```

where:

<0-9> specifies the global variable ID. You can define up to nine global variables i.e. var.globa.0 var.global.1.

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

Table 3-12: Configuration Examples using Variables in Message Manipulations Table

Message Type	Condition	Action Subject	Action Type	Action Value
invite		var.global.0	Modify	'Custom UA'
invite	param.message.sdp.rtpmode=='sendrecv'	var.call.src.1	Modify	'1'
invite .response. 200	var.call.dst.0=='1'	param.message.sdp.rtpmode	Modify	'sendonly'

3.8 Random Characters

The following syntax shows how to specify random letter characters in the range *a* to *z* in the Message Manipulations table.

Syntax:

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

where:

- **<n>** is the number of random letter characters you wish to specify in the range *a* to *z*.

The following syntax shows how to specify random letter 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 letter and/or numeric characters you wish to specify in the range 0 to z.

The following syntax shows how to specify random numbers 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.
- **<m>** specifies the end value of the range of the random numbers that you wish to specify.

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

Table 3-13: Configuration Examples using Random Letters & Numeric Characters in Message Manipulations Table

Message Type	Action Subject	Action Type	Action Value
invite.request	header.myrandomString	Add	Rand.string.56.A.Z
invite.response	header.NumberaAndChars	Add	Rand.string.12.0.z
invite.response.4xx	header.myrandomNmber	Add	Rand.number.50.100

3.9 Regular Expressions

This following syntax shows how to specify regular expressions in the Message Manipulations table.

Syntax:

```
<regular expression>
```

where:

- **<Regular expression>** is used as part of the value in a condition and contains a regular expression.

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

Table 3-14: Configuration Examples Regular Expressions in Message Manipulations Table

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
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

Reader's Notes

4 Summary of Typical Examples

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

Table 4-1: Typical Examples For Message Manipulation Sets

Message Type	Condition	Action Subject	Action Type	Action Value
invite.request	param.message.sdp.address=='flowers.com'	header.diversion	Add	'<sip:WeSellFlowers@p4.isp.com>;reason=time-of-day'
info.response	header.request-uri.methodtype=='488'	header.request-uri.methodtype	Modify	'503'
info.response.180		header.request-uri.methodtype	Modify	'183'
invite.request	header.expires.time < '88888'	header.organisation	Add	'audiocodes'
register.request		header.contact.param.newparam	Add	'newValue'
subscribe.response		header.remote-party-id.0.partytype	Modify	'2'
invite.response		header.from.param.nasty	Delete	
any		header.user-agent	Modify	'TelcoA'

Reader's Notes

5 Detailed SIP Header Syntax

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

Table 5-1: 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	
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 (GRUU) contact	header.contact.gruucontact	
	Enables GRUU	header.contact.isgruu	
	Name	header.contact.name	
	Parameter	header.contact.param	
	URL: ▪ type (enum): SIP (1), Tel (2), Fax (3), SIPS (4) ▪ host (port / name) ▪ mhost ▪ userphone (0/1) ▪ loosephone (0/1) ▪ user (string) ▪ transporttype ▪ param	header.contact.url.<url type>	header.contact.url.user
Cseq	Header itself	header.cseq	
	Number	header.cseq.num	header.cseq.num=='1'
	Type	header.cseq.type	
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 – yes / no	header.diversion.screen	
	URL (see URL for Contact header)	header.diversion.url	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
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	
	Parameter	header.from.param	header.from.param.p1
	Tag	header.from.tag	
	URL (see URL for Contact header)	header.from.url	header.from.url.user != '654'
History-Info	Header itself	header.history-info	
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 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-uri.param	
	URL (see URL 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 for Contact header)	header.p-called-party-id.url	
P-Charging-	Header itself	header.p-charging-vector	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
Vector			
P-Preferred-Identity	Header itself	header.p-preferred-identity	
	Name (string)	header.p-preferred-identity.name	
	URL (see URL for Contact header)	header.p-preferred-identity.url	
Privacy	Header itself	header.privacy	
	Privacy types: ▪ none ▪ header ▪ session ▪ user ▪ critical ▪ identity ▪ history	header.privacy.privacy.<type>	header.privacy.privacy.user
Proxy-Require	Header itself	header.proxy-require	
	SIP Capabilities: ▪ earlymedia ▪ reliableresponse ▪ timer ▪ earlysession ▪ privacy ▪ replaces ▪ history ▪ unknown ▪ gruu ▪ resourcepriority ▪ targetdialog ▪ sdpanat	header.proxy-require.<capability>	header.proxy-require.earlymedi
Reason	Header itself	header.reason	
	Reason types: ▪ Reason ▪ Cause ▪ text	header.reason.reason.<type>	header.reason.reason.reason
	MLPP: Type: Preemption (0), MLPP (1) cause	header.reason.mlpp	
Referred-By	Header itself	header.referred-by	
	Parameter	header.referred-by.param	header.referred-by.param.p1
	URL (see URL for Contact header)	header.referred-by.url	header.referred-by.url.host
Refer-To	Header itself	header.refer-to	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
Remote-Party-ID	Header itself	header.remote-party-id	
	Counter	header.remote-party-id.counter	
	Name	header.remote-party-id.name	
	Number Plan: ▪ ISDN (1) ▪ Data (3) ▪ Telex (4) ▪ National (8) ▪ Private (9) ▪ Reserved (15)	header.remote-party-id.numberplan	
	Number Type	header.remote-party-id.numbertype	
	Parameter	header.remote-party-id.param	
	Privacy (see Privacy header for description)	header.remote-party-id.privacy	
	Reason types: ▪ Busy ▪ Immediate ▪ No Answer	header.remote-party-id.reason.<type>	header.remote-party-id.reason.busy
	Screen – Yes / No	header.remote-party-id.screen	
	Screen Indicator types (enum): ▪ User Provided ▪ User Passed ▪ User Failed ▪ Network Provided	header.remote-party-id.screenind.<type>	
	URL (see URL for Contact header)	header.remote-party-id.url	
Request-URI	Header itself	header.request-uri	
	Method	header.request-uri.method	
	Method Type	header.request-uri.methodtype	header.request-uri.methodtype == '5'
	URI	header.request-uri.uri	
	URL (see URL for Contact header)	header.remote-party-id.url	header.request-uri.url.user == '101'
Require	Header itself	header.require	
	SIP Capabilities (see SIP Capabilities for Proxy-Require header)	header.require	header.require.earlymedia
Resource-Priority	Header itself	header.resource-priority	
	Namespace	header.resource-	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
		priority.namespace	
	RPriority	header.resource-priority.rpriority	
Retry-After	Header itself	header.retry-after	
	Time	header.retry-after.time	
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	
	Time	header.session-expires.time	
Subject	Header itself	header.subject	
Supported	Header itself	header.supported	
	SIP Capabilities (see SIP Capabilities for Proxy-Require header)	header.supported.<capability>	header.supported.path
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 for Contact header)	header.to.url	header.to.url.userphone
Unsupported	Header itself	header.unsupported	
	SIP Capabilities (see SIP Capabilities for Proxy-Require header)	header.unsupported.<capability>	header.unsupported.path
User-To-User and X-UserToUser	Header itself	header.x-usertouser	
	User-to-User Descriptor	header.x-usertouser.user2user	
	Protocol Descriptor (PD)	header.x-usertouser.pd	

SIP Header	Attribute to Manipulate	Manipulation Syntax	Example
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	
	Transport type: ▪ UDP (0) ▪ TCP (1) ▪ TLS (2) ▪ SCTP (3)	header.via.transporttype	header.via.0.transporttype == '0'
Warning	Header itself	header.warning	
Unknown headers	Header itself	header.<unknown header name>	header.color

5.1 Wildcarding 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"

Note: The wildcard does not remove the following headers:

- Request-Uri
- Via
- From
- To
- Callid
- Cseq
- Contact

5.2 Message Manipulation using SDP Conditions

You can configure message manipulation rules based on user-defined SDP conditions.

The device supports the following SDP condition syntax:

- **Source IP Address:** You can manipulate the source IP address in the SDP. For example, you can configure a manipulation rule to add a Diversion header to incoming INVITE messages if the SDP contains a specific IP address, or a prefix or suffix of this IP address.
  - **param.message.sdp.ip suffix** '10.10'
  - **param.message.sdp.ip prefix** '10.132'
  - **param.message.sdp.ip==**'10.33.37.78'
- **RTP mode:** You can manipulate the RTP mode using the following condition:
  - **param.message.sdp.rtpmode**

  Possible values include the following:
  - sendonly
  - sendrecv
  - inactive
- **Origin IP Address:** Using the origin IP address (in the SDP "o=" line):
  - **param.message.sdp.originaddress**

  Possible values include any IP address.
- **Port:** First audio active media port number (i.e., port number greater than 0) in the "m=" field of the SDP body:
  - **sdp.port**
- **IP address:** IP address of the first active media (port greater than 0). 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, then the IP address is taken from the global "c=" field (the "c=" field at the top of the SDP):
  - **sdp.address**

Below are manipulation examples using SDP conditions:

- **Example 1:** Copy 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, as follows:

Message Type	Action Subject	Action Type	Action Value
invite.request	header.custom-rtp-address	Add	param.message.sdp.ip
invite.request	header.custom-rtp-port	Add	param.message.sdp.port

- **Example 2:** Changes the RTP mode to sendonly if the SDP "c=" line address is 0.0.0.0:

Message Type	Condition	Action Subject	Action Type	Action Value
reinvite.request	param.message.sdp.ip == '0.0.0.0'	param.message.sdp.rtpmode	Modify	'sendonly'

- **Example 3:** Changes the SDP "c=" line to the same address as the "o=" line:

Message Type	Action Subject	Action Type	Action Value
-	param.message.sdp.ip	Modify	param.message.sdp.originaddress

- **Example 4:** Condition the RTP mode:

Message Type	Condition	Action Subject	Action Type	Action Value
invite	param.message.sdp.rtpmode =='sendrecv'	var.call.src.1	Modify	'1'
invite. response.200	var.call.dst.0== '1'	param.message.sdp.rtpmode	Modify	'sendonly'

- **Example 5:** The manipulation rule example below 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, i.e., 10.33. The IP address is contained in the "c=" line of the SDP (e.g., "c=IN IP4 10.33.37.75"). The table below shows the example configuration:

Parameter	Rule Index 1	Rule Index 2
Message Type	invite	invite
Condition	param.message.sdp.ip =='10.33.37.78'	param.message.sdp.ip prefix '10.33'
Action Subject	header.diversion	header.diversion
Action Type	Add	Add
Action Value	<sip:12345@p4.isp.com>;reason=no-answer	<sip:12345@p4.isp.com>;reason=no-answer

You can configure several such manipulation rules and then apply them per IP Group using the 'Inbound Message Manipulation Set' parameter.

Note: This feature is applicable only to the SBC application.

5.3 Using 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).

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.

Below are a few examples 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@company244.com
      ```

  - Manipulation rule:

Index	Condition	Action Subject	Action Type	Action Value
1	header.to regex (<.*)(**[3-**4]**[0-**9])(.*)(\d)@(.*>)	header.to	Modify	$1+'2'+$2+ $3+'@'+$5

  - **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=XINPYDPROEOREGE
IHUHF
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:

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

Note: 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 on SDP:**
  - Manipulation required: To change the packet period in the SDP.
  - Manipulation rule:

Index	Condition	Action Subject	Action Type	Action Value
1	body.sdp regex (.*)(a=ptime:20)(.*)	body.sdp	Modify	$1+'a=ptime:10'+$3

  - **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.

Reader's Notes

AudioCodes

Quick Reference Guide

www.audiocodes.com