ADTRAN Operating System (AOS) Manipulating SIP Headers and Messages in AOS

Configuration Guide

6AOSCG0026-29E

June 2020

To the Holder of this Document

Trademark Information

Disclaimer of Liability

Revision History

Rev D	October 2018	Initial release of document in this format. Document updated to include the Call-ID variables, a string matching function, and a renumbering command that were introduced in AOS firmware R12.1.0 release, the random number generation system variable that was introduced in AOS firmware R12.3.0 release, and the intercept polices that were introduced in AOS firmware R12.4.0 release.
Rev E	June 2020	Updated document format and made minor corrections.

Table of Contents

1. Overview of SIP Header and Message Manipulation in AOS

This configuration guide describes the use and manipulation of Session Initiation Protocol (SIP) headers and message bodies for inbound and outbound SIP messages on ADTRAN Operating System (AOS) products. SIP headers can be modified in SIP messages using the command line interface (CLI). This guide provides an overview of how SIP header manipulation operates, the steps necessary to configure SIP header manipulation, configuration examples, and troubleshooting steps.

Beginning with AOS firmware release R10.1.0, the SIP header manipulation feature was introduced for AOS products. This feature allows the manipulation of both SIP headers and message bodies in SIP transmissions, based on configurable rules. These rules can be applied to both outbound and inbound messages, and can be used to match SIP headers, modify existing SIP headers or the body of SIP messages, add SIP headers, remove SIP headers, and store variable information. In addition, message manipulation rules can use regular expressions to modify SIP headers and message bodies. One of the benefits of using SIP header and message manipulation is that the feature can give you enhanced control over the behavior of SIP traffic on your AOS device, as well as help solve interoperability issues between the AOS device and other products.

SIP Header Manipulation Operation

SIP header manipulation is achieved by creating a header manipulation rule (HMR) policy, a set of HMR rules, and applying those rules to the HMR policy. The policy is then applied to a SIP trunk, to all SIP traffic in the AOS device, to SIP traffic sent or received by a SIP proxy user, or to SIP traffic sent or received by a SIP proxy server. The HMR policies can be applied to either inbound or outbound SIP traffic. In addition, HMR variables can be created, which store text used in header manipulation. These variables can be either public or private, indicating their scope of access. Public variables are accessible and shared by all policies. Private variables exist within an instance of a policy.

The following are the basic configuration steps needed to create SIP header manipulation in AOS:

1. Create an HMR rule set.
2. Create HMR rule(s) for the rule set.
3. Specify each HMR rule’s action (match, modify, add, remove).
4. Optionally, configure variables for the HMR rule.
5. Create an HMR policy.
6. Assign the HMR rule set to the appropriate HMR policy.
7. Apply the HMR policy to all SIP traffic, SIP proxy users, SIP proxy servers, or a SIP trunk.

2. Hardware and Software Requirements and Limitations

SIP header and message manipulation is a feature that is available only on AOS products running AOS firmware R10.1.0 or later. This feature is available on the products as outlined in the *AOS Feature Matrix*, available online at https://supportcommunity.adtran.com.

- In AOS firmware release R11.6.0, system variables and variable manipulation were introduced.
- In AOS firmware release R11.10.0, support for conditional logic within HMR message rule configurations was introduced.
- In AOS firmware release R12.1.0, Call-ID variables, a string matching function, and a renumbering command were introduced.
- In AOS firmware release R12.3.0, a random number generation system variable was introduced.
- In AOS firmware release R12.4.0, intercept policies were introduced.

- HMR rules can be applied selectively to inbound or outbound SIP traffic.
- When using regular expressions, they must be enclosed by a pair of /. Option flags for case insensitivity (*i*), global matching (*g*), and multiline (*m*) can follow the second /. For more information about regular expressions, refer to *Appendix A: Regular Expressions on page 42*.
- Sequence numbering for HMR rules in the rule set and for message rules within the HMR rule are defaulted to an increment of 10 (10, 20, 30, etc.). All rules are processed in sequence.
- Match commands apply to the message and must resolve as true for action commands to be processed. The match-value parameter of an action command is only in effect for that command. If multiple match commands are present within a single HMR rule, all must resolve to be true.
- HMR rule sets are referenced by HMR policies, and are treated as inbound or outbound according to the policy. Rule set names must be unique for the device. In addition, HMR message rule names are not referenced outside of the rule set, therefore, they must only be unique within a rule set.
- Variables can be referenced within patterns and strings using the naming convention *%scope.var-name%*. All string and pattern values can reference variable names. When variables are used, the rule parser parses the string or pattern and replaces any variable names with the corresponding value. If a variable name is entered incorrectly, the string is treated as a literal string rather than a variable name. If the variable name is properly entered, but it does not exist, an empty value is used.
- AOS CLI uses quotation marks, question marks, and spaces as native elements or delimiters in its parsing. To include any of those in a regular expression pattern as a literal string, use the hex representation instead (for example, use **\x22** for quotation marks). To invoke the match of the pattern **/.*"Unavailable".*/**, you would use **/.*\x22Unavailable\x22.*/**. To use a question mark as a repetition quantifier, enter it as **%q%**. When specifying strings, enclosing quotes are optional unless the pattern includes spaces.

3. Configuring SIP Manipulation Using the CLI

SIP header and message manipulation is achieved in AOS products using the CLI. The following are the basic configuration steps necessary to implement SIP header and message manipulation.

Step 1. Access the CLI

To access the CLI on your AOS unit, follow these steps:

1. Boot up the unit.
2. Telnet to the unit (**telnet** <*ip address*>). For example:

```
telnet 10.10.10.1
```

NOTE

If during the unit's setup process you have changed the default IP address (10.10.10.1), use the configured IP address.

3. Enter your user name and password at the prompt.

NOTE

*The AOS default user name is **admin** and the default password is **password**. If your product no longer has the default user name and password, contact your system administrator for the appropriate user name and password.*

4. Enable your unit by entering enable at the prompt as follows:

```
>enable
```

5. If configured, enter your Enable mode password at the prompt.
6. Enter the unit's Global Configuration mode as follows:

```
#configure terminal
(config)#
```

Step 2. Create an HMR Rule Set

Once you have connected to your AOS device, the next step in configuring SIP header and message manipulation is to create an HMR rule set. An HMR rule set is a named collection of one or more sequenced message rules. When a rule set is applied to a message, all matching message rules are processed in sequence.

To create an HMR rule set and enter the rule set's configuration mode, enter the **hmr rule-set** <*name*> command from the Global Configuration mode. The <*name*> parameter is the name given to the HMR rule set. Names must be unique for each rule set you configure. Multiple rule sets can be created by entering the command multiple times. To create the HMR rule set, and enter its configuration mode, enter the command as follows:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#
```

Step 3. Create HMR Message Rules for the Rule Set

After creating the HMR rule set, and entering the rule set's configuration, your next step is to configure the necessary HMR rules. A message rule is a collection of one or more header commands, which determine the types of SIP headers to act upon, and the action to be taken. When a message rule is applied to a SIP message, all matching header commands are processed. Message rules are processed in the order determined by the sequence number of the message rule within the rule set. When multiple rules are applied to a message, the results of each rule are applied before the next rule is evaluated and applied.

Each message rule is named, and can be configured to apply to all SIP messages, or only to SIP requests or responses. To create an HMR message rule, and enter the rule's configuration mode, enter the **message-rule** <*name*> **[message-type [request | response | any]] [**<*sequence number*>**]** command from the rule set's configuration mode. The <*name*> parameter specifies the name of the message rule. Message rule names must be unique within the HMR rule set. The optional **message-type** parameter specifies whether the rule is applied to SIP **request** or **response** messages, or to both (**any** keyword). The optional <*sequence number*> parameter specifies the sequence number given to the message rule, which determines the order in which the rules are processed. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**. Entering the command from the HMR Rule Set Configuration mode prompt as follows creates a message rule named **Rule1** that applies to **any** SIP message type and has a sequence number of **5**, and it also enters the rule's configuration mode:

```
(config-rule-set-Set1)#message-rule Rule1 message-type any 5
(config-msg-rule-Rule1)#
```

Step 4. Specify the HMR Message Rule's Action

After creating an HMR message rule, you can specify the rule's action(s). HMR message rules can be used to add new SIP headers to the message, remove SIP headers from the message, modify existing SIP headers in the message, and modify the existing body in a SIP message. These actions are completed on SIP headers/messages that match specified criteria.

When multiple rules are applied to a message, the results of each rule are applied before the next rule is evaluated and applied.

In addition, HMR message rules can be used to manipulate public and private variables. Configuration of variables is discussed in more detail in *Step 8. Optional Configuration: Variables on page 19*. The following sections discuss the various actions available for HMR message rules, including the support for conditional logic configurations (outlined in *Using Conditional Logic with HMR Message Rules (Optional) on page 12*).

Matching Headers

HMR message rules can be used to match specified SIP headers in a SIP message using the **match header** <*header*> **[match-value** <*pattern*>**]** command from the HMR Message Rule Configuration mode. Match commands allow you to specify conditions that must be true in order for the message rule to be processed. If a **match header** command is present in the message rule's configuration, the message rule is processed only if the **match header** resolves as true. Multiple match header commands can be present, but all of them must resolve as true for the message rule to be processed. The <*header*> parameter of the command indicates the SIP header to be used for matching. *Table 1* outlines the SIP header choices for rule matching.

Table 1. Available SIP Header Choices for Rule Matching

SIP Headers		
Accept-Contact	P-Asserted-Identity	Supported
Allow	P-Preferred-Identity	To
Allow-Events	Record-Route	Via
Call-Id	Refer-To	<*name*> (specifies another header not listed here)
Contact	Referred-By	
Content-Encoding	Reject-Contact	
Content-Length	Replaces	
Content-Type	Request-Disposition	
Diversion	Route	
Event	Session-Expires	
From	Sip-Req-Uri	
Identity	Sip-Status-Line	
Identity-info	Subject	

NOTE

The SIP header can also be expressed in compact form in a SIP message. Compact forms of the header name will match the full SIP header name.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*.

If both the header and a match pattern are specified, the indicated SIP header must be present in the SIP message and must contain the specified pattern for the rule to resolve as true. If only the SIP header is specified, the SIP message must contain the specified SIP header for the rule to resolve as true.

To specify that the HMR rule match SIP headers based on a specific header and a specified pattern (for example, matching REFER requests), enter the command from the HMR Message Rule Configuration mode as follows:

```
(config-msg-rule-Rule1)#match header sip-req-uri match-value /^REFER.*/
(config-msg-rule-Rule1)#
```

Matching SIP Message Bodies

HMR message rules can be used to match specified body contents of SIP messages using the **match body [match-value** <*pattern*>**]** command from the HMR Message Rule Configuration mode. Match commands allow you to specify conditions that must be true in order for the message rule to be processed. If a **match body** command is present in the message rule's configuration, the message rule is processed only if the **match body** resolves as true. The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching. The <*pattern*> parameter can be a regular expression or a text string. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If the match pattern is provided, the statement will only evaluate to true if the message body contains the match pattern specified. If no match pattern is specified, the message body is matched unconditionally, if present.

To specify that the HMR rule match SIP messages based on the contents of the message body, enter the **match body** command from the HMR Message Rule Configuration mode as follows:

```
(config-msg-rule-Rule1)#match body match-value "/G729/"
(config-msg-rule-Rule1)#
```

Matching Variables

SIP messages can also be matched based on variables. Variables enable the storage of text that is used in header manipulation. You can have a private or public variable, and can match messages based on both variable types. To match based on a public variable, enter the command **match public-variable** <*variable*> **[match-value** <*pattern*>**]** from the HMR Message Rule Configuration mode. To match based on a private variable, enter the command **match private-variable** <*variable*> **[match-value** <*pattern*>**]** command. The <*variable*> parameter specifies a previously configured variable (detailed in *Step 8. Optional Configuration: Variables on page 19*), and is expressed in the format **%private.VariableName%**. The **match-value** <*pattern*> parameter specifies the pattern to be used for matching, in addition to the variable. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If the pattern is specified, the variable must contain the specified value in order for the rule to be processed. If the pattern is not specified, the match resolves as true if the variable is defined.

To match based on a private variable, enter the command as follows:

```
(config-msg-rule-Rule1)#match private-variable myUnitNumber match-value "253"
(config-msg-rule-Rule1)#
```

Adding Headers

You can also add headers to SIP messages using SIP header manipulation. To add a new header to the SIP message, enter the **add header** <*header*> **position [first | if-not-present | last] new-value** <*value string*> [<*sequence number*>] command from the HMR Message Rule Configuration mode. The <*header*> parameter specifies the SIP header type, that you would like to add to the SIP message. Available headers are outlined in *Table 1 on page 8*. The **position** keyword specifies where the new header is added. You can add the header as the first of the specified type (**first**), add the header only if it is not already present (**if-not-present**), or add the header as the last of the specified header type within the message (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The **new-value** parameter specifies the value to assign to the new header, and is expressed as a text string, for example, **“localID=unit195”**. The optional <*sequence number*> parameter specifies the sequence number given to the message rule, which determines the order in which the rules are processed. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

To add a new SIP header to the SIP message, enter the add header command as follows from the HMR Message Rule Configuration mode:

```
(config-msg-rule-Rule1)#add header Proprietary1 position first new-value
     localID=unit195;remoteID=unit272
(config-msg-rule-Rule1)#
```

Removing Headers

SIP headers can also be removed from SIP messages using SIP header manipulation. Headers are removed from messages by entering the **remove header** <*header*> **position [all | first | first-match | last] [match-value** <*pattern*>**] [**<*sequence number*>**]** command from the HMR Message Rule Configuration mode. The <*header*> parameter specifies the SIP header type that you would like to remove from the SIP message. Available headers are outlined in *Table 1 on page 8*. The **position** keyword specifies the position in the SIP message of the header to be removed.

You can remove all matching headers of the specified header type (**all**), remove the first header of the specified header type if there is a match (**first**), remove the first matching header of the specified header type regardless of its position within the message (**first-match**), or remove the last header of the specified header type is there is a match (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and filters the SIP headers. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If both the match pattern and SIP header are specified, the indicated header is removed if it contains the match pattern. If only the header type is specified, the SIP header is removed unconditionally. The optional <*sequence number*> parameter specifies the sequence number given to the message rule, which determines the order in which the rules are processed. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

To remove a SIP header from the SIP message, enter the remove header command as follows from the HMR Message Rule Configuration mode:

```
(config-msg-rule-Rule1)#remove header Proprietary1 position first
(config-msg-rule-Rule1)#
```

Modifying Headers

In addition to adding or removing SIP headers, you can also use SIP header manipulation to modify existing SIP headers within the SIP message using the **modify header** <*header*> **position [all | first | first-match | last] [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]** command from the HMR Message Rule Configuration mode. The <*header*> parameter specifies the SIP header type that you would like to modify in the SIP message. Available headers are outlined in *Table 1 on page 8*. The **position** keyword specifies the position in the SIP message of the header to be modified. You can modify all matching headers of the specified header type (**all**), the first header of the specified header type if there is a match (**first**), the first matching header of the specified header type regardless of its position within the message (**first-match**), or the last header of the specified header type is there is a match (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and filters the SIP headers. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If both the match pattern and SIP header are specified, the indicated header is modified if it contains the match pattern. If only the header type is specified, the SIP header is modified unconditionally. The **new-value** <*pattern*> parameter specifies the value to be assigned to the header, and can include buffers captured with the optional **match-value** parameter. The <*pattern*> parameter can be a regular expression or a text string. The optional <*sequence number*> parameter specifies the sequence number given to the message rule, which determines the order in which the rules are processed. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

To modify a SIP header in the SIP message, for example, to add <> to Contact headers that do not contain them, enter the command from the HMR Message Rule Configuration mode as follows:

```
(config-msg-rule-Rule1)#modify header contact position all match-value
     /([^<]*)(sip:.*@.*[^;])(.*)/ new-value /\1<\2>\3/
(config-msg-rule-Rule1)#
```

Modifying SIP Messages

In addition to modifying SIP headers, SIP header manipulation also allows you to manipulate the body of existing SIP messages. To modify SIP messages, enter the **modify body [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]** command from the HMR Message Rule Configuration mode. The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and additionally provides capture buffers for the message body replacement if needed. The <*pattern*> parameter can be a regular expression or a text string. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If the match pattern is provided, the content of the message is only modified if the message contains the match pattern specified. If no match pattern is specified, the message body is modified unconditionally. The **new-value** <*pattern*> parameter specifies the value to be assigned to the message body, and can include buffers captured with the optional **match-value** <*pattern*> parameter. The **new-value** <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. The optional <*sequence number*> parameter specifies the sequence number given to the message rule, which determines the order in which the rules are processed. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

When the message body is modified, SIP HMR automatically adjusts the Content-Length header of the message to match the length of the new value.

To modify the body of a SIP message, enter the **modify body** command from the HMR Message Rule Configuration mode as follows:

```
(config-msg-rule-Rule1)#modify body new-value pattern1
(config-msg-rule-Rule1)#
```

Using Conditional Logic with HMR Message Rules (Optional)

In HMR, the configuration of rule components includes specifying the following:

- Specifying the rule set (*Step 2. Create an HMR Rule Set on page 7*)
- Specifying the HMR rules for the rule set (*Step 3. Create HMR Message Rules for the Rule Set on page 7*). This includes specifying the match criteria for the rule, or the rule actions.
- Specifying the rule actions (*Step 4. Specify the HMR Message Rule's Action on page 7*). These actions include modifying, adding, or removing headers, setting public or private variables, or modifying the body of SIP messages.

The basics of this configuration are outlined in the following template, which also describes their location within the configuration:

```
rule-set
   message-rule
       match criteria (includes match header, body, and private or public variables)
       actions (includes modifying, removing, or adding headers, setting public or private variables, and
           modifying the body of SIP messages)
```

In AOS firmware release R11.10.0, two new constructs were added to HMR rule configuration: the conditional logic based on **if** criteria and the conditional logic based on **for** criteria for a specified SIP header. These new parameters fit into HMR configuration as follows:

```
rule-set
   message-rule
   match criteria (includes match header, body, and private or public variables)
       actions (includes modifying, removing, or adding headers, setting public or private variables, and
           modifying the body of SIP messages)
   if (conditional logic block)
   for header (conditional logic block)
```

These new commands allow HMR rules to be executed based on whether certain criteria are returned as true or false. They are configured within the HMR Message Rule configuration mode using the **if** and **for** commands. These two commands signal the beginning of a logic block in which other criteria can be configured and used. For example, the **if** command is used to specify that an action is taken when the logical condition of the rule returns as true. Once the **if** command is entered, you can then specify the criteria which should be evaluated, and the actions to be taken if the statement is true. Each logic block ends using the appropriate **end** command. The specifics of these commands are outlined in *Table 2 on page 13*.

The **if** conditional logic block takes the following form (entered from the (config-msg-rule-RULE)# prompt):

```
if
  (condition block)
    (actions)
  else (optional)
```

(actions)
end-if

The **for header** logic block takes the following form (entered from the (config-msg-rule-RULE)# prompt):

for header *<header>*
(condition block) (optional)
(actions)
end-for

The configuration of HMR rules using these parameters appears as follows:

```
(config)#hmr rule-set SET1
(config-rule-set-SET1)#message-rule RULE1
(config-msg-rule-RULE1)#match header identity
(config-msg-rule-RULE1)#if
(config-msg-rule-RULE1)#compare “teststring1” equal “teststring2”
(config-msg-rule-RULE1)#modify body new-value PATTERN1
(config-msg-rule-RULE1)#end-if
```

Table 2 describes the **if** and **for header** commands.

Table 2. If and For Header Conditional Logic Commands

Condition Logic Begin Commands	Condition Logic End Commands	Description
if [*<sequence number>***]**	**end-if [***<sequence number>***]**	Specifies that all the rules or actions of the conditional logic block are executed if the condition block returns true. The optional *<sequence number>* parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **if** command is entered, actions or conditions should then be specified. Once all conditions and actions for the **if** (or **else**) logic have been entered, use the **end-if** command to signify the end of this logical block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Table 2. If and For Header Conditional Logic Commands (Continued)

Condition Logic Begin Commands	Condition Logic End Commands	Description
else [<*sequence number*>**]**		Optional parameter for the **if** condition block. Specifies an action alternative to those specified in the **if** command is executed if the condition block returns false. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **else** command is entered, actions or conditions should then be specified. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
for header <*header*> **[**<*sequence number*>**]**	**end-for [**<*sequence number*>**]**	Configures a loop of conditional logic or specific actions for the specified SIP header. Available headers are outlined in *Table 1 on page 8*. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **for header** command is entered, actions or conditions and actions should then be specified. Once all conditions and actions for the **for header** logic have been entered, use the **end-for** command to signify the end of this logical block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Condition Blocks and Their Components

The condition block in the **if** and **for header** commands represent the configuration of conditional logic that is evaluated before executing the actions in the conditional logic block. Condition blocks are composed of a condition block beginning, tests, and a condition block ending. Condition block tests include comparing values and evaluating expressions. The following sections describe the condition block and its components.

Condition Block Components

Conditional logic blocks are blocks of conditional logic that follow the **if** or **for** commands in HMR rule configuration. Conditional logic blocks take the following format:

conditional block (entered using the **if** or **for** command)
  condition block tests begin (optional; **all** by default)
    condition block tests
  condition block end (must enter if condition logic begin is specified)

Condition Block Beginnings and Endings

Condition block beginnings include the **not**, **all**, or **any** commands. Using a condition block beginning is optional; by default, **all** is implied. Condition block endings include **end-not**, **end-all**, and **end-any**. An ending must be specified if a beginning is specified. By default, the implied condition block ending is **end-all**. *Table 3* describes the commands used for beginning and ending condition blocks.

Table 3. Conditional Logic Block Beginning and Ending Commands

Condition Block Begin Commands	Condition Block End Commands	Description
all [<*sequence number*>**]**	**end-all [**<*sequence number*>**]**	Specifies that the rules or actions of the conditional logic block are executed only if all tests in the condition block evaluate to true. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **all** command is entered, conditions should then be specified. Once all conditions for the **all** logic have been entered, use the **end-all** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
any [<*sequence number*>**]**	**end-any [**<*sequence number*>**]**	Specifies that the rules or actions of the conditional logic block are executed if any conditions in the condition block are met. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **any** command is entered, conditions should then be specified. Once all conditions for the **any** logic have been entered, use the **end-any** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
not [<*sequence number*>**]**	**end-not [**<*sequence number*>**]**	Specifies that all the rules or actions of the conditional logic block are executed after a test for the negative of the configured logical block conditions. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **not** command is entered, conditions should then be specified. Once all conditions for the **not** logic have been entered, use the **end-not** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Condition Block Tests

Conditional block tests are tests specified within the **if** or **for** conditional logic blocks. In the conditional logic block format, condition block tests are located between the condition block beginning and end:

conditional logic block (entered using the **if** or **for** command)
  condition block begin (optional; **all** by default)
    condition block tests
  condition block end (must enter if conditional logic begin is specified)

Condition block tests are composed of either compare statements, which include comparison and evaluation, or additional nested conditional logic blocks (such as those outlined in *Table 3 on page 15*). You can enter one or more of these in any combination.

Compare statements are used to compare two expressions according to a logical operator, and return either true or false. Compare statements can be used to perform text, string, or integer comparisons, and follow this format:

```
compare <expression> <logical operator> <expression> [<sequence number>]
```

OR

```
int-compare <integer expression> <logical operator> <integer expression> [<sequence
     number>]
```

Expressions and integer expressions can be text strings, regular expressions, variables, or variable functions. If the text value contains a space, it must be enclosed in quotation marks; otherwise, quotation marks are optional for text values. Integer expressions return integer values for HMR processing. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. The available logical operators used in these commands are outlined in *Table 4*.

Table 4. Logical Operators for Compare Commands

Logical Operator	Description
equal	Specifies the condition is returned true if the first expression is equal to the second expression.
greater-equal	Specifies the condition is returned true if the first expression is greater than or equal to the second expression.
greater	Specifies the condition is returned true if the first expression is greater than the second expression.
less-equal	Specifies the condition is returned true if the first expression is less than or equal to the second expression.
less	Specifies the condition is returned true if the first expression is less than the second expression.
not-equal	Specifies the condition is returned true if the first expression is not equal to the second expression.

For example, you can configure the actions within the **if** conditional logic block of HMR rule **RULE1** to be applied if the expression comparison conditions are resolved to true. In this case, the **if** conditional logic block is configured to perform integer comparison on two variables, and then to modify the SIP message body if the condition evaluates to true:

```
(config)#hmr rule-set SET1
(config-rule-set-SET1)#message-rule RULE1
(config-msg-rule-RULE1)#if
(config-msg-rule-RULE1)#int-compare %private.variabletest1% equal
     %private.variabletest2%
(config-msg-rule-RULE1)#modify body new-value PATTERN1
(config-msg-rule-RULE1)#end-if
```

The **evaluate** command, when used in a conditional logic block, can be used to trigger an evaluation of a specified expression. This returned value from the evaluation is used to test the true or false requirement for behavior within the conditional logic block. The **evaluate** logical command is expressed as follows:

```
evaluate <expression> [<sequence number>]
```

The expressions evaluated with this command can be a regular expression, a text string, or a variable. Text strings that include spaces must be enclosed in quotation marks; otherwise, quotation marks are optional. This commands executes the expression, and returns a true if the execution was successful, or a false if it was not. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**.

Conditional Logic Configuration Considerations

All components of the HMR rule configuration, whether actions or conditional logic blocks, can be configured in almost any combination. Each HMR rule configuration can include one or more action or conditional logic blocks in any combination.

With the conditional logic additions, it is possible that there could be duplicate commands within the SIP HMR configuration. For that reason, when using the **no** version of a command, you must enter a sequence number. By default, sequence numbering occurs in increments of **10**.

Once you have configured the actions, match criteria, and optional conditional logic for the HMR rules, you can continue to *Step 5. Creating an HMR Policy* to finish the HMR configuration.

Step 5. Creating an HMR Policy

After you have created an HMR rule set, and configured the message rules and behaviors, you must create an HMR policy. The HMR policy is a named collection of one or more rule sets, and the policy is used to apply the rule sets to specific SIP traffic, SIP proxy user traffic, SIP proxy server traffic, or trunks. Create the HMR policy using the **hmr policy** <*name*> command from the Global Configuration mode. The <*name*> parameter is the name given to the policy. For example, to create the HMR policy **MyPolicy1**, enter the command as follows:

```
(config)#hmr policy MyPolicy1
Configuring new HMR policy MyPolicy1
(config-policy-MyPolicy1)#
```

Step 6. Apply HMR Rule Sets to an HMR Policy

After creating the HMR policy in the Global Configuration mode, you must then apply the relevant rule sets to the policy. Rule sets are added to the policy using the **rule-set** <*name*> **[**<*sequence number*>**]** command from the HMR Policy Configuration mode. The <*name*> parameter is the name of the previously created rule set that you want to apply to the policy. Multiple rule sets can be added to a single policy. When multiple rule sets are applied to a message, the results of each rule set are applied before the next rule set is evaluated and applied. The optional <*sequence number*> parameter specifies the sequence number given to the rule set within the policy. This number determines the sequencing for applying the various rule sets in the policy. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

To add a rule set to an HMR policy, enter the rule-set command from the HMR Policy Configuration mode as follows:

```
(config-policy-MyPolicy1)#rule-set Set1
(config-policy-MyPolicy1)#
```

Step 7. Apply the HMR Policy

Once the HMR policy is configured with the proper rule sets, the policy must be applied to SIP traffic in the unit. Policies do not take effect until they are applied to one of the following: all SIP traffic, SIP trunk traffic, SIP proxy user traffic, or SIP proxy server traffic. Policies can be applied to multiple types of SIP traffic. When multiple rules or policies are applied to a message, the results of each rule are applied before the next rule is evaluated and applied.

HMR policies can be applied to inbound or outbound SIP messages. Policies used for SIP proxy traffic are always classified as user or server when they are applied. SIP proxy user traffic is traffic to or from devices, such as phones, behind the SIP proxy. This will usually be the local area network (LAN) side of the network.

SIP proxy server traffic is traffic to or from the SIP server. This will usually be the wide area network (WAN) side of the network.

NOTE

For inbound traffic, global policies are applied before trunk or SIP proxy policies. For outbound traffic, global policies are applied after trunk or SIP proxy policies.

The following commands are used to apply SIP HMR policies to SIP traffic.

NOTE

Because HMR application does have an impact on the performance of your AOS unit, you should be as restrictive as possible in your application of HMR policies. For example, if you only need to manipulate outbound traffic on a SIP trunk, apply an outbound policy to the trunk, rather than applying a global outbound policy.

Applying an HMR Policy to All SIP Traffic

To apply an HMR policy to all SIP traffic on the AOS unit, enter the **sip hmr** <*policy name*> **[in | out]** command from the Global Configuration mode prompt. The <*policy name*> parameter specifies the HMR policy you are applying. In addition, you must specify whether the policy is applied to incoming SIP traffic or outgoing SIP traffic on the device by using either the **in** or **out** keywords.

To add the HMR policy MyPolicy1 to the AOS unit for all inbound SIP traffic, enter the command as follows:

```
(config)#sip hmr MyPolicy1 in
(config)#
```

This policy will now apply the configured manipulation rules to the inbound SIP traffic on the AOS device.

Applying an HMR Policy to a SIP Trunk

To apply an HMR policy to a SIP trunk, you must enter the trunk's configuration mode and add the policy to the trunk using the **hmr** <*policy name*> **[in | out]** command. The <*policy name*> parameter specifies the HMR policy you are applying. In addition, you must specify whether the policy is applied to incoming SIP traffic on the trunk, or outgoing SIP traffic on the trunk by using either the **in** or **out** keywords.

To add the HMR policy **MyPolicy1** on the SIP trunk **T01** for all inbound SIP traffic, enter the command as follows:

```
(config)#voice trunk t01
(config-T01)#hmr MyPolicy1 in
(config-T01)#
```

This policy will now apply the configured manipulation rules to the inbound SIP traffic on the trunk.

Applying an HMR Policy to SIP Proxy User Traffic

To apply an HMR policy to the SIP proxy traffic to or from devices (such as phones) behind the SIP proxy, enter the **sip proxy hmr user** <*policy name*> **[in | out]** command from the Global Configuration mode prompt. The <*policy name*> parameter specifies the HMR policy you are applying. In addition, you must specify whether the policy is applied to incoming SIP proxy user traffic, or outgoing SIP proxy user traffic by using either the **in** or **out** keywords.

To add the HMR policy **MyPolicy1** for all inbound SIP proxy user traffic, enter the command as follows:

```
(config)#sip hmr user MyPolicy1 in
```

This policy will now apply the configured manipulation rules to the inbound SIP proxy traffic for all proxy users.

Applying an HMR Policy to SIP Proxy Server Traffic

To apply an HMR policy to the SIP proxy traffic to or from proxy servers, enter the **sip proxy hmr server** <*policy name*> **[in | out]** command from the Global Configuration mode prompt. The <*policy name*> parameter specifies the HMR policy you are applying. In addition, you must specify whether the policy is applied to incoming SIP proxy server traffic, or outgoing SIP proxy server traffic by using either the **in** or **out** keywords.

To add the HMR policy **MyPolicy1** for all inbound SIP proxy server traffic, enter the command as follows:

```
(config)#sip hmr server MyPolicy1 in
(config)#
```

This policy will now apply the configured manipulation rules to the inbound SIP proxy traffic for all proxy servers.

Step 8. Optional Configuration: Variables

In addition to the basic configuration for SIP header manipulation (configuring rule sets, message rules and actions, and HMR policies), you can also configure public, private, and Call-ID variables. Variables allow for storage of text that is used in header manipulation. These variables can be used when referencing a pattern for matching or manipulation. Public variables are those that are visible to all HMR rules within the AOS unit, and private variables are those that are visible only within a single HMR policy. Private variables are modified within the HMR Message Rule Configuration mode, and when configured, are defined for the current policy only. Call-ID variables are similar to public variables in that they're available to all policies, but their value is limited in scope to the current Call-ID. All variables are offset in a set of **%** characters when entered, and public variables are noted with the **public** keyword, for example, **%public.MyGroupNumber%**. Private variables are noted with the **private** keyword, for example, **%private.MyUnitNumber%**. Call-ID variables are noted with the callid keyword, for example, **%callid.MySequenceNumber%**.

Variables are created within the HMR Message Rule Configuration mode, and public variables can also be configured from the Global Configuration mode, but all are accessed using message rules. Variables can then be referenced by patterns within the message rule configuration. The following commands are used to manipulate variables.

Private Variables

To set the value of a private variable (only visible within a single HMR policy), enter the **set private-variable** <*variable name*> **[body] [header** <*header*> **position [first | first-match | last]] [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]** from the HMR Message Rule Configuration mode. Using this command will specify a variable to be modified and used by the message rule. The variable to be modified is specified using the <*variable name*> parameter. Private variables are referenced in the format **%private.variablename%**. The optional **body** parameter specifies that the optional match pattern applies to the contents of the SIP message. The optional **header** parameter specifies whether the optional match pattern applies to the contents of a SIP header or to the existing contents of the variable. Available headers are outlined in *Table 1 on page 8*. In addition, when using the SIP header to specify the data source, you must also specify the position of the header within the message that you want to use by entering the **position** keyword. You can select the first header of the specified header type if there is a match (**first**), the first matching header of the specified header type regardless of its position within the message (**first-match**), or the last header of the specified header type if there is a match (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and filters based on the specified SIP header or the current contents of the variable. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If both the match value and SIP header are specified, the indicated variable is modified if the header is present and contains the match value. If only the **match-value** parameter is specified, the match pattern is applied to the variable's current value. The **new-value** <*pattern*> parameter specifies the value to be assigned to the variable, and can include buffers captured with the optional **match-value** <*pattern*> parameter. The **new-value** <*pattern*> parameter can be a regular expression or a text string. The optional <*sequence number*> parameter specifies the sequence number given to the rule, which determines the order in which the rules are processed within the message rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

For example, to set a private variable called **unitID** that holds the value of the unit ID parameter from the **target-unit** header of the SIP message, enter the command from the HMR Message Rule Configuration mode as follows:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#message-rule Rule1 message-type any
(config-msg-rule-Rule1)#set private-variable unitID header target-unit position first
     match-value /.*unitID=(.*)/ new-value /\1/
```

Public Variables

Unlike private variables, public variables can optionally be specified in the Global Configuration mode, and then modified within the HMR Message Rule Configuration mode, if necessary, or configured in the HMR Message Rule Configuration mode explicitly. To set the value of a public variable from the Global Configuration mode, enter the **hmr set public-variable** <*variable name*> **new-value** <*pattern*> command from the Global Configuration mode prompt. The <*variable name*> parameter specifies the variable to be set. The <*pattern*> parameter specifies the new value to be used by the variable and is a text string.

To set the value of a public variable, enter the command from the Global Configuration mode prompt as follows:

```
(config)#hmr set public-variable paiTest new-value nomatch
(config)#
```

To set or modify a public variable while in the HMR Message Rule Configuration mode, enter the **set public-variable** <*variable name*> **[body] [header** <*header*> **position [first | first-match | last]] [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]** from the HMR Message Rule Configuration mode. Using this command will specify a variable to be modified and used by the message rule. The variable to be modified is specified using the <*variable name*> parameter. Public variables are referenced in the format **%public.variablename%**. The optional **body** parameter specifies that the optional match pattern applies to the contents of the SIP message. The optional **header** parameter specifies whether the optional match pattern applies to the contents of a SIP header or to the existing contents of the variable. Available headers are outlined in *Table 1 on page 8*. In addition, when using the SIP header to specify the data source, you must also specify the position of the header within the message that you want to use by entering the **position** keyword. You can select the first header of the specified header type if there is a match (**first**), the first matching header of the specified header type regardless of its position within the message (**first-match**), or the last header of the specified header type if there is a match (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and filters based on the specified SIP header or the current contents of the variable. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If both the match value and SIP header are specified, the indicated variable is modified if the header is present and contains the match value. If only the **match-value** parameter is specified, the match pattern is applied to the variable's current value. The **new-value** <*pattern*> parameter specifies the value to be assigned to the variable, and can include buffers captured with the optional **match-value** <*pattern*> parameter. The **new-value** <*pattern*> parameter can be a regular expression or a text string. The optional <*sequence number*> parameter specifies the sequence number given to the rule, which determines the order in which the rules are processed within the message rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

For example, to modify the public variable **paiTest** (set in the previous example), enter the command from the HMR Message Rule Configuration mode as follows:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#message-rule Rule1 message-type any
(config-msg-rule-Rule1)#set public-variable paiTest new-value match 20
(config-msg-rule-Rule1)#
```

Call-ID Variables

To set the value of a Call-ID variable (scope limited to the current Call-ID), enter the **set callid-variable** <*variable name*> **[body] [header** <*header*> **position [first | first-match | last]] [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]** command from the HMR Message Rule Configuration mode. Using this command will specify a variable to be modified and used by the message rule. The variable to be modified is specified using the <*variable name*> parameter. Call-ID variables are referenced in the format **%callid.variablename%**. The optional **body** parameter specifies that the optional match pattern applies to the contents of the SIP message. The optional **header** parameter specifies whether the optional match pattern applies to the contents of a SIP header or to the existing contents of the variable. Available headers are outlined in *Table 1 on page 8*. In addition, when using the SIP header to specify the data source, you must also specify the position of the header within the message that you want to use by entering the **position** keyword. You can select the first header of the specified header type if there is a match (**first**), the first matching header of the specified header type regardless of its position within the message (**first-match**), or the last header of the specified header type if there is a match (**last**).

NOTE

Multiple headers of the same type can occur in SIP messages, and therefore the specified position of the header can determine whether a match occurs. For more information, refer to Appendix B on page 44.

The optional **match-value** <*pattern*> parameter specifies the pattern to be used for matching, and filters based on the specified SIP header or the current contents of the variable. The <*pattern*> parameter can be a regular expression or a text string, and can reference variable names. For more information about using regular expressions, refer to *Appendix A: Regular Expressions on page 42*. If both the match value and SIP header are specified, the indicated variable is modified if the header is present and contains the match value. If only the **match-value** parameter is specified, the match pattern is applied to the variable's current value. The **new-value** *<pattern>* parameter specifies the value to be assigned to the variable, and can include buffers captured with the optional **match-value** *<pattern>* parameter. The **new-value** *<pattern>* parameter can be a regular expression or a text string. The optional *<sequence number>* parameter specifies the sequence number given to the rule, which determines the order in which the rules are processed within the message rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**. For example, to set a Call-ID variable called **myCSeq** that holds the value of the the **CSeq** header of the SIP message, enter the command from the HMR Message Rule Configuration mode as follows:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#message-rule Rule1 message-type any
(config-msg-rule-Rule1)#set callid-variable myCSeq header cseq position first
     match-value /(.*)/ new-value /\1/
```

System Variables

System variables were introduced in AOS R11.6.0 and provide predefined variables to access system-level information. System variables are referenced in the format **%system.variablename%** or **%system.variablename%.function(<parameters>)**. AOS provides the following system variables:

```
|| Variable || Optional Functions ||
| date | format(<format string>) |
| random | range(<min>, <max>) |
```

The **format string** for the **format()** function for the **date** system variable is specified as follows in *Table 5*:

Table 5. Date System Variable Format String

Symbol	Meaning	Presentation	Example
G	Era designator	Text	AD
y	Year	Number	1996
M	Month in year	Text & Number	July & 07
d	Day in month	Number	10
h	Hour in am/pm (1-12)	Number	12
H	Hour in day (0-23)	Number	0
m	Minute in hour	Number	30
s	Second in minute	Number	55
S	Millisecond	Number	978
E	Day in week	Text	Tuesday
D	Day in year	Number	189
a	am/pm marker	Text	PM
k	Hour in day (1-24)	Number	24

Table 5. Date System Variable Format String (Continued)

Symbol	Meaning	Presentation	Example
K	Hour in am/pm (0-11)	Number	0
z	Time zone	Text	CST

The minimum value for the **range()** function for the **random** system variable is - 2,147,483,647 and the maximum is 2,147,483,647.

Variable Manipulation

In AOS firmware release R11.6.0, additional manipulation capability is included for both public and private variables. These features allow you to perform additional operations on the content of both public and private variables. These operations include (but are not limited to) appending to, deleting from, or replacing variable content as well as searching or comparing variable content. To perform these operations on variable content, manipulative actions are specified in the variable's pattern definition.

Variables are defined using the following commands:

```
set private-variable <variable name> [body] [header <header> position [first |
    first-match | last]] [match-value <pattern>] new-value <pattern> [<sequence
    number>]
```

AND

```
set public-variable <variable name> [body] [header <header> position [first |
    first-match | last]] [match-value <pattern>] new-value <pattern> [<sequence
    number>]
```

When the variable is defined with one of these commands, a new value (**new-value** *<pattern>*) or a match value (**match-value** *<pattern>*) can also be defined. These values, specified in the *<pattern>* parameter, are expressed as a regular expressions or text strings. To perform additional operations on the content of the variable, you can specify an action in the *<pattern>* parameter of the new or matched value. In this case, the *<pattern>* parameter is not just a regular expression or text string, but rather a combination of the variable name and the operation you want to perform on the variable's content; it is expressed as *<variable name>.<function>*, so that, for example, **set private-variable messageCount new-value MATCH** becomes **set private-variable messageCount new-value /%private.messageCount%.add(1)**. The *<variable name>* parameter in this instance is the variable that contains the content to be manipulated, and the *<function>* parameter is the manipulative action to be applied to that content. The functions that can be performed on the variable content are outlined in *Table 6*.

Table 6. Variable Manipulation Action Pattern Definitions

Function	Action	Parameters	Returns
.add(<*value*>**)**	Adds the specified value to the variable.	<*value*> specifies the value to add to the variable.	The variable once the specified value is added.
.append(<*string*>**)**	Appends additional characters at the end of the current value.	<*string*> defines the characters to append.	The new variable string with the appended information.

Table 6. Variable Manipulation Action Pattern Definitions (Continued)

Function	Action	Parameters	Returns
.compare(<*string*>**)**	Compares the variable string to the specified value.	<*string*> specifies the string to which the variable is compared. For example, **%public.myVar%compare('test string')** compares the current value of **myVar** to the parameter **test string**.	An integer result of **0** (indicates the values are equal), **1** (indicates the variable is greater than the value entered), or **-1** (indicates the variable is less than the value entered).
.erase(<*position*>,<*count*>**)**	Erases characters from the variable string.	<*position*> optionally specifies the position of the first character to be erased. <*count*> optionally specifies the number of characters to be erased.	The new variable string after the specified characters have been removed.
.find(<*string*>, <*start*>**)**	Finds content within the variable string by searching for the first occurrence of the specified sequence.	<*string*> specifies the set of characters to be used in the search. <*start*> optionally specifies the position of the first character to be considered in the search.	The integer position of the searched-for content within the variable string.
.find_first_not_of(<*string*>, <*start*>**)**	Finds the absence of specified characters within the variable string by searching the string for the first character that does not match the specified characters.	<*string*> specifies the set of characters to be used in the search. <*start*> optionally specifies the position of the first character to be considered in the search.	The integer position of the first non-matching character within the variable string.
.find_first_of(<*string*>, <*start*>**)**	Finds characters in the variable string by searching for the first character that matches any of the specified characters.	<string> specifies the set of characters to be used in the search. <start> optionally specifies the position of the first character to be considered in the search.	The integer position of the first character match within the variable string.
.find_last_not_of(<*string*>, <*start*>**)**	Finds non-matching characters in the variable string by searching the string for the last character that does not match the characters specified.	<string> specifies the set of characters to be used in the search. <start> optionally specifies the position of the last character to be considered in the search.	The integer position of the first character in the variable string that does not match the search criteria.

Table 6. Variable Manipulation Action Pattern Definitions (Continued)

Function	Action	Parameters	Returns
.find_last_of(*<string>, <start>***)**	Finds characters in the variable string by searching for the last character that matches any of the specified characters.	<string> specifies the characters to be used in the search. <start> optionally specifies the position of last character to be considered in the search.	The integer position of the last character that matches the search criteria.
.insert(*<position>, <string>***)**	Inserts specified information into the variable string.	<position> specifies the insertion point. The new content is inserted before this point. <string> specifies the characters to be inserted.	The variable string with the new information inserted.
.int_compare(*<value>***)**	Compares strings as though they were integers.	<value> specifies the string to which the variable is compared. This is an integer value.	An integer result of **0** (indicates strings are equal), **1** (indicates the entered variable is greater than the value entered), or **-1** (indicates the entered variable is less than the value entered).
.length()	Finds the length of the variable string.	N/A	The length of the string, in bytes.
.match(*<match pattern>***[,** *<substitute pattern>***])**	Performs regular expression string matching and substitution on a string expression.	*<match pattern>* specifies the regular expression pattern to match against the target string expression. *<substitute pattern>:* specifies the regular expression pattern to apply as substitution against the target string expression.	An integer result of **1** (indicates if the match succeeds), or **0** (indicates if the match fails).
.replace(*<position>, <count>, <string>***)**	Replaces a portion of a variable string. The replacement occurs at the defined *<position>*.	*<position>* specifies the position of the first character to be replaced. *<count>* specifies the number of characters to be replaced (length of replacement). *<string>* specifies the replacement characters.	The variable string with the replacement complete.
.rfind(*<string>, <start>***)**	Finds the last occurrence of the specified characters within the variable string.	<string> specifies the characters for which to search. <start> optionally specifies the position of the last character to be considered as the beginning of a match.	The integer position within the variable string of the first character of the last match.

Table 6. Variable Manipulation Action Pattern Definitions (Continued)

Function	Action	Parameters	Returns
.sub(<*value*>**)**	Subtracts the specified value from variable.	<value> specifies the value to subtract from the variable.	The variable once the specified value is subtracted.
.substr(<*position*>**,** <*count*>**)**	Generates a substring from the variable string.	<position> optionally specifies the starting position from which the substring is generated. <count> optionally specifies the number of characters to include in the substring.	The generated substring.

Variable Manipulation Considerations

In variable manipulation, integer and string <*value*> parameters are interchangeable, and type interpretation is based on their usage. For example, performing an **.int_compare()** action on a variable will treat it as integer, whereas performing a **.compare()** action on a variable will treat it as a string. The integer value of a non-numeric string is **0**.

In addition, it is possible for an operation that returns a string to have another operation called on it. For example, entering **%myStringValue%.append('Adding text at end.').length()** will both append the text and return the length of the entire string. If this value is assigned to a variable, only the length is assigned. Functions that appear to be modifying the variable, such as **.append()** or **.erase()**, do not actually modify it when the function is called with a period (**.**), as in **%someString%.append('APPENDTEXT')**. To modify the variable with this operation, use the number sign (**#**), as in **%someString%#append('APPENDTEXT')**.

Functions must include parentheses, as these contain the operational parameters. Some parameters are optional; however, if two parameters are available, to input the second parameter the first must be specified. For example, entering **erase()** simply erases the string, entering **erase(3)** erases from the third position to the end of the string, and entering **erase(3,4)** erases four characters of the string starting at the third position. You cannot specify four characters are erased (second parameter) if the position (first parameter) is not specified. If no parameters are needed, such as with the function **length()**, the parentheses are still required.

NOTE

In the AOS CLI, double quotes within a string value are not allowed. Therefore, quoted strings are denoted with single quotes, for example, ***%myString%.find('teststring')****.*

Variable Manipulation Examples

The following example increments a user variable that is used to track SIP INVITEs:

```
(config)#hmr rule-set INVITECOUNT
(config-rule-set-INVITECOUNT)#message-rule TestAdd1
(config-msg-rule-TestAdd1)#match header sip-req-uri match-value /INVITE/
(config-msg-rule-TestAdd1)#set public-variable inviteCount new-value
     /%public.inviteCount%.add(1)/
(config-msg-rule-TestAdd1)#exit
(config-rule-set-INVITECOUNT)#message-rule TestAdd2
(config-msg-rule-TestAdd2)#match public-variable match-value /10/
(config-msg-rule-TestAdd2)#add header MyInviteNotification position first new-value
     "INVITE count just hit 10!"
```

4. HMR Intercept

In AOS R12.4.0, HMR Intercept was added. HMR Intercept is a mechanism that allows standard HMR policies to be used to alter the flow of selected SIP requests. HMR Intercept policies are given first access to inbound requests. A policy may be used to generate a response to a request, block processing of a request, or modify a request before other SIP agents within the device gain control of it. A policy's rules will be evaluated to determine whether one or more rules match a give SIP request. If a match occurs, all rules that match within the policy will be applied to the traffic in the same way that inbound/outbound policies are applied. Additionally, the HMR Intercept action assigned to the policy will be applied to the request.

The respond and block actions, when invoked, take ownership of the SIP request. No other SIP agent within the device will be given access to the request.

The modify action, when invoked, will apply the assigned policy to the request but will not take ownership of it.

Multiple actions may be specified for an HMR Intercept policy. Additionally, multiple policy assignments may be made for a given action. Matching actions are applied in sequence number order. If a respond or block policy matches a request, ownership is given to that policy and policy selection is halted. If a modify policy matches a request, the policy is applied and policy selection continues.

HMR Intercept Policy Types

Respond Policy

A respond action policy accepts a matching request and sends an appropriate response. The configuration of the respond action may specify the response code and/or text to be used. If no response code is configured, the default of **200** is used. The generated response will be modified by applicable rules that are included in the policy.

The rule-set for this type of policy should include a message-rule of type request that contains the necessary match rules to match the inbound request. A second message-rule of type response can be included if modification of the generated response is necessary.

Block Policy

A block action policy accepts a matching request and terminates the request. No response is generated.

The rule-set for this type of policy should include a message-rule of type request that contains the necessary match rules to match the inbound request. No other rules are necessary since matching will cause the request to be ignored.

Modify Policy

A modify action policy applies the rules included in the policy. This action is taken prior to other SIP agents within the device gaining access to the request. Once the policy is applied, normal processing continues.

The rule-set for this type of policy should include a message-rule of type request that contains the necessary match rules to match the inbound request as well as the HMR rules to modify the message as needed.

Configuring HMR Intercept

The CLI commands for HMR Intercept provide a means to enable/disable HMR Intercept, assign policies to it, and define actions for those policies.

Step 1: Enable HMR Intercept

Use the **hmr intercep**t command to enable HMR Intercept policy matching.

```
(config)#hmr intercept
```

Step 2: Assign HMR Intercept Policies

Multiple actions may be specified for a single HMR Intercept policy. Additionally, multiple policy assignments may be made for a given action. Matching actions are applied in sequence number order.

Assign a policy to a Response action:

```
(config-hmr-intercept)#respond <name> [code <number>][text <"response text">] [<seq
    number>]
```

Assign a policy to a Block action:

```
(config-hmr-intercept)#block <name> [<seq number>]
```

Assign a policy to a Modify action:

```
(config-hmr-intercept)#modify <name> [<seq number>]
```

Syntax Description

<name>	Specifies the name of the HMR policy to which this action is to be assigned
code *<number>*	Optional. Specifies the SIP response code to be sent in the response message. Valid range is **1** to **999**. Default value is **200**.
text *<"response text">*	Optional. Specifies the text to be sent in the response message.
<seq number>	Optional. Specifies the sequence number assigned to this action. Valid range is **1** to **99,999**. The default value is a sequence number that will place the new action at the end of the list of configured actions.

Step 3: Activate HMR Intercept

Use the **no shut** command to activate HMR Intercept policies:

```
(config-hmr-intercept)#no shut
```

NOTE

*If changes are made to HMR Intercept policies, you must issue the **shut** command followed by the **no shut** command for the changes to take effect.*

HMR Intercept Example

Consider the following example configuration that contains HMR intercept policies:

```
hmr rule-set BlockOtherRegisters
  message-rule MatchBlock message-type request 10
    match header sip-req-uri match-value REGISTER
!
hmr rule-set RespondToRegisterFromChicago
  message-rule MatchRegister message-type request 10
    match header sip-req-uri match-value REGISTER
    match header contact match-value PlantA
  message-rule ModifyResponse message-type response 20
    add header PlantHeader position first new-value "Plant A" 10
!
hmr rule-set RespondToRegisterFromNJ
  message-rule MatchRegister message-type request 10
    match header sip-req-uri match-value REGISTER
    match header contact match-value PlantB
  message-rule ModifyResponse message-type response 20
    add header PlantHeader position first new-value "Plant B" 10
```

```
!
hmr rule-set ModifyNotify
  message-rule ModifyNotify message-type request 10
    match header sip-req-uri match-value NOTIFY
    add header MyHeader position first new-value "New Header" 10
!
hmr policy RespondToRegisterFromChicago
  rule-set RespondToRegisterFromChicago 10
!
hmr policy RespondToRegisterFromNJ
  rule-set RespondToRegisterFromNJ 10
!
hmr policy BlockOtherRegisters
  rule-set BlockOtherRegisters 10
!
hmr policy ModifyNotify
  rule-set ModifyNotify 10
!
hmr intercept
  respond RespondToRegisterFromChicago code 200 10
  respond RespondToRegisterFromNJ code 200 20
  block BlockOtherRegisters 30
  modify ModifyNotify 40
  no shut
```

Assume this configuration is in effect for the following example scenarios:

1. A REGISTER request with the text “PlantA” in the Contact header is received.

   This request would match the first intercept action. The remaining actions would not be examined. The generated response would then have the custom header “PlantHeader” containing the value “Plant A” added to it.

2. A REGISTER request with the text “PlantB” in the Contact header is received.

   This request would match the second intercept action. The remaining actions would not be examined. The generated response would then have the custom header “PlantHeader” containing the value “Plant B” added to it.

3. A REGISTER request with the text “PlantC” in the Contact header is received.

   This would not match either of the first two intercept actions. The third action matches and triggers a block of the request so that no further action will be taken by the device.

4. A NOTIFY request is received.

   This would match the fourth intercept action. A custom header MyHeader containing the value “New Header” is added to the request. The request is then subject to normal processing by the device.

5. SIP Header Manipulation Configuration Examples

The example scenarios contained within this section are designed to enhance understanding of SIP header manipulation configurations on AOS products. The examples describe some of the common real-world applications of HMR. All configurations provided in this section use the command line interface (CLI).

NOTE

The configuration parameters entered in these examples are sample configurations only. These applications should be configured in a manner consistent with the needs of your particular network. CLI prompts have been removed from the configuration examples to provide a method of copying and pasting configurations directly from this configuration guide into the CLI. These configurations should not be copied without first making the necessary adjustments to ensure they will function properly in your network.

Example 1: Moving Orbit Parameter for SIP Stack

When a SIP REFER request is received in the format **sip:user@host.com;orbit=x**, the SIP stack interprets this as **<sip:user@host.com>;orbit=x**. One of the actions the SIP proxy must perform in processing these requests is to move the orbit parameter so that the form is **<sip:user@host.com;orbit=x>**. The following sample HMR configuration can accomplish this.

```
hmr rule-set myFirstRuleSet
 message-rule myOrbitFix message-type request
   match header sip-req-uri match-value /^REFER.*/
   modify header refer-to position first-match match-value
    /(.*)>(;orbit=[^;]+)(.*)/new-value /\1\2>\3/
!
hmr policy myGlobalInboundPolicy
 rule-set myFirstRuleSet
!
sip hmr myglobalInboundPolicy in
!
```

Example 2: Multiple HMR Manipulations

In the following example, multiple actions are performed. First, a non-standard SIP header is added to all outbound SIP requests on a SIP trunk. Second, a parameter is added to the Contact header of all outbound SIP requests on a SIP trunk. Third, the message on outbound 481 responses is changed to **Route Unavailable**.

```
hmr rule-set mySecondRuleSet
 message-rule myAddProprietaryAddHeader message-type request
   add header Proprietary1 position if-not-present new-value
           localID=unit195;remoteId=unit272
   modify header Contact position first match-value /([^@]+)(.*)/ new-value
    /\1;tgrp=2565551000;trunk-context=domain.com\2/
 message-rule my481ResponseChange message-type response
   modify header sip-status-line position first match-value "/(.*481 ).*/" new-value
    "/\1Route Unavailable/"
!
hmr policy myTrunkOutboundPolicy
 rule-set mySecondRuleSet
!
!
voice trunk t01
 hmr myTrunkOutboundPolicy out
```

Example 3: Adding <> to Contact Headers

In the following example, HMR is used to add <> to Contact headers that do not contain them. It is also used to replace all occurrences of **;** with **,** in Allow-Events headers and to remove a **/** at the end of Content-Type headers.

```
hmr rule-set myThirdRuleSet
 message-rule mySyntaxDoctor message-type any
   modify header contact position first match-value /([^<]*)(sip:.*@.*[^;])(.*)/
    new-value /\1<\2>\3/
   modify header allow-events position first match-value /([^;,]*)[;,]/g new-value
    /\1,/
   modify header content-type position first match-value /(.*)\/$/ new-value /\1/
!
hmr policy myTrunkInboundPolicy
 rule-set myThirdRuleSet
!
voice trunk t01
 hmr myTrunkInboundPolicy in
!
```

Example 4: Adding Parameters to Headers

In the following example, a particular SIP message from a user is expected to have the custom header **target-unit**. The value of the unit ID parameter is to be added as the URI parameter *unit* in the SIP URI of the contact header of the message containing the **target-unit** header. This message rule is applied to any SIP message processed that contains a target-unit header. A private variable called **unitId** is modified to hold the value of the unit ID parameter, and the URI in the Contact header is then modified to contain the URI parameter *unit* with the specified value.

```
hmr rule-set UnitIDRuleSet
 message-rule myUnitIdFixup
   match header target-unit
   set private-variable unitId header target-unit position first match-value
    /.*unitID=([^;]*)/ new-value /\1/
   modify header contact position first-match match-value /(.*<.*)(>.*)/ new-value
    1;unit=%private.unitId%\2/
   remove header target-unit position first
!
```

Example 5: Adding P-Asserted-Identity for ANI Matching

In this example, the task is to add a P-Asserted-Identity header to SIP requests if the From header user does not match a list of automatic number identification (ANI) numbers configured in the AOS device. This task is completed by defining a message rule action that initializes a private variable for a no-match condition, defining message rule actions to change the value of the private variable to indicate a match condition if the message's From header matches a given ANI number, and to configure a second message rule to add the P-Asserted-Identity header if the private variable contains the original initialized value (no match condition).

```
hmr rule-set paiRuleset
 message-rule paiRule1 message-type request 10
   set private-variable paiTest new-value nomatch 10
   set private-variable paiTest header from position first match-value
    /.*sip:2565550051.*/ new-value /match/ 20
   set private-variable paiTest header from position first match-value
    /.*sip:2565550052.*/ new-value /match/ 30
 message-rule paiRule2 message-type request 20
   match private-variable paiTest match-value nomatch
```

```
    add header p-asserted-identity position if-not-present new-value 2565551000 10
!
hmr policy myPolicy1
 rule-set paiRuleset 10
!
voice trunk T01 type sip
 hmr myPolicy1 out
!
```

Example 6: Removing Port Number from Request URI

In the following example, the task is to remove the port number from the Request URI on incoming SIP messages.

```
hmr rule-set portRemove
 message-rule portRemoveRule message-type request 10
 modify header sip-req-uri position first match-value "/([A-Z]+ sip:.*):\d+(.*)/"
    new-value /\1\2/ 10
!
hmr policy myPolicy1
 rule-set portRemove 10
!
sip hmr myPolicy1 in
!
```

Example 7: Using HMR to Initiate RTP Loopback based on Originating Number

In the following example, HMR is used to initiate Realtime Transfer Protocol (RTP) loopback based on the originating number (caller ID) rather than the terminating number (called number). This can be useful to assign a finite group of phone numbers to initiate RTP loopbacks rather than using a phone number. To use HMR in this manner, you must first configure a loopback account on the AOS device. This part of the configuration is not in this example; for more information about configuring loopback accounts, refer to the configuration guide *Configuring the AOS Voice Loopback Account*, available online at https://supportcommunity.adtran.com. Once you have configured the loopback account, it must be enabled on both the voice trunk and loopback user. Next, HMR is configured to modify the SIP message body to include two additional lines in the Session Description Protocol (SDP) (**a=loopback:rtp-pkt-loopback** and **a=loopback-source:0**). Lastly, the created HMR policy is applied to the voice trunk.

In this example, note the match header statements used in the HMR policy. Without the match header, this configuration can cause every call into the AOS device to become a loopback call. The match header tells the AOS device that a specific number that is calling is to be routed back where it came from. In addition, note the modify body commands and their match value parameters. The modify body commands are necessary because of the way the body of the message is arranged. Each modify body command and match value parameter is terminated with a carriage return and line feed expression so that the lines are added correctly.

```
!
voice trunk T01
 media-loopback
!
voice loopback 1234
 media-loopback
!
hmr rule-set mySet1
 message-rule MyMessage message-type request 10
 match header from match-value /.*0052.*/
 modify body match-value /((.*\r\n)*)/ new-value /\1a=loopback:rtp-pkt-loopback/ 10
 modify body match-value /((.*\r\n)*)/ new-value /\1a=loopback-source:0/ 20
!
```

```
hmr policy myPolicy1
 rule-set mySet1 10
!
voice trunk T01
 hmr myPolicy1 in
!
```

6. SIP Header Manipulation Command Summary

The following table summarizes the commands used in SIP header manipulation.

Table 7. SIP Header Manipulation Command Summary

Prompt	Command	Description
(config)#	**hmr rule-set** *<name>*	Creates an HMR rule set and enters the rule set's configuration mode.
(config-rule-set-set1)#	**message-rule** *<name>* **[message-type [request | response | any]] [***<sequence number>***]**	Creates a message rule for the rule set. Optionally specifies the message type to which the rule applies, and optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config-msg-rule-rule1)#	**match header** *<header>* **[match-value** *<pattern>***]**	Specifies the message rule matches a specified SIP header. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable.
(config-msg-rule-rule1)#	**match body [match-value** *<pattern>***]**	Specifies the message rule for matching SIP message body contents. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable.
(config-msg-rule-rule1)#	**match public-variable** *<variable>* **[match-value** *<pattern>***]**	Specifies the message rule matches a specified public variable. Public variables are referenced in the format **%public.VariableName%**. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable.
(config-msg-rule-rule1)#	**match private-variable** *<variable>* **[match-value** *<pattern>***]**	Specifies the message rule matches a specified private variable. Private variables are referenced in the format **%private.VariableName%**. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable.
(config-msg-rule-rule1)#	**add header** *<header>* **position [first | if-not-present | last] new-value** *<value string>* [*<sequence number>***]**	Specifies that the message rule adds a SIP header and indicates the position in which to add the header. The **new-value** parameter specifies the value to assign to the new header, and is expressed as a text string. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config-msg-rule-rule1)#	**remove header** *<header>* **position [all | first | first-match | last] [match-value** *<pattern>***] [***<sequence number>***]**	Specifies that the message rule removes a SIP header and indicates in which position the header that is to be removed resides. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config-msg-rule-rule1)#	**modify header** *<header>* **position [all | first | first-match | last] [match-value** *<pattern>***] new-value** *<pattern>* **[***<sequence number>***]**	Specifies that the message rule modifies the specified header and in which position the specified header resides. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable. The **new-value** parameter specifies the new value to assign to the header, which can be a regular expression, text string, or variable. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config-msg-rule-rule1)#	**modify body [match-value** *<pattern>***] new-value** *<pattern>* **[***<sequence number>***]**	Specifies that the message rule modify the body of a SIP message. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable. The **new-value** parameter specifies the new value to assign to the message body, which can be a regular expression, text string, or variable. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config-msg-rule-rule1)#	**if [***<sequence number>***]**	Specifies that the rules or actions of the conditional logic block are executed if the condition block returns true. The optional *<sequence number>* parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **if** command is entered, actions or conditions should then be specified. Once all conditions and actions for the **if** (or **else**) logic have been entered, use the **end-if** command to signify the end of this logical block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config-msg-rule-rule1)#	**else [<*sequence number*>]**	Optional parameter for the **if** condition block. Specifies an action alternative to those specified in the **if** command is executed if the condition block returns false. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **else** command is entered, actions or conditions should then be specified. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**for header** <*header*> **[<*sequence number*>]**	Configures a loop of conditional logic or specific actions for the specified SIP header. Available headers are outlined in *Table 1 on page 8*. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **for header** command is entered, actions or conditions should then be specified. Once all conditions and actions for the **for header** logic have been entered, use the **end-for** command to signify the end of this logical block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**all [<*sequence number*>]**	Specifies that the rules or actions of the condition block are executed only if all tests in the condition block evaluate to true. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **all** command is entered, conditions should then be specified. Once all conditions for the **all** logic have been entered, use the **end-all** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config-msg-rule-rule1)#	**any [**<*sequence number*>**]**	Specifies that the rules or actions of the conditional logic block are executed if any conditions in the condition block are met. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **any** command is entered, conditions should then be specified. Once all conditions for the **any** logic have been entered, use the **end-any** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**not [**<*sequence number*>**]**	Specifies that all the rules or actions of the conditional logic block are executed after a test for the negative of the configured logical block conditions. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Once the **not** command is entered, conditions should then be specified. Once all conditions for the **not** logic have been entered, use the **end-not** command to signify the end of this condition block. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**compare** <*expression*> <*logical operator*> <*expression*> **[**<*sequence number*>**]**	Compares two expressions according to a logical operator, and returns either true or false. Compare expressions are text strings, enclosed in quotation marks. The optional <*sequence number*> parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. The available logical operators are outlined in *Table 4 on page 16*. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config-msg-rule-rule1)#	**int-compare** *<integer expression> <logical operator> <integer expression>* **[***<sequence number>***]**	Compares two integer expressions according to a logical operator, and returns an integer value. Integer compare expressions are text strings, regular expressions, variables, or variable functions. Text strings are enclosed in quotation marks. The optional *<sequence number>* parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. The available logical operators are outlined in *Table 4 on page 16*. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**evaluate** *<expression>* **[***<sequence number>***]**	Evaluates the validity of an expression by executing the expression. Expressions are regular expressions, text strings, variables, or variable functions. Text strings are enclosed in quotation marks. The command returns true if the execution was successful, false if it was not. The optional *<sequence number>* parameter determines the order in which the rule is processed. Valid range is **1** to **99999**. By default, sequence numbering occurs in increments of **10**. Use the **no** form of this command to remove the command. When using the **no** form of the command, you must specify a sequence number.
(config-msg-rule-rule1)#	**set private-variable** *<variable name>* **[body] [header** *<header>* **position [first | first-match | last]] [match-value** *<pattern>***] new-value** *<pattern>* **[***<sequence number>***]**	Sets the value of a private variable. The optional **body** parameter specifies that matching occurs based on the body contents of the SIP message. The optional **header** parameter specifies that if a data capture buffer is used, the data source is the value of the header specified and the location of the particular header. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable. In addition, variable patterns can be used for manipulation, as outlined in *Variable Manipulation on page 23*. The **new-value** parameter specifies the value to assign to the variable, which can be a regular expression or text string. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config-msg-rule-rule1)#	**set public-variable** <*variable name*> **[body] [header** <*header*> **position [first | first-match | last]] [match-value** <*pattern*>**] new-value** <*pattern*> **[**<*sequence number*>**]**	Sets the value of a public variable and specifies that the message rule use this variable. The optional **body** parameter specifies that matching occurs based on the body contents of the SIP message. The optional **header** parameter specifies that if a data capture buffer is used, the data source is the value of the header specified and the location of the particular header. Optionally specifies a pattern used for matching. Patterns can be a regular expression, text string, or variable. In addition, variable patterns can be used for manipulation, as outlined in *Variable Manipulation on page 23*. The **new-value** parameter specifies the value to assign to the variable, which can be a regular expression or text string. Optionally assigns a sequence number to the rule. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config-msg-rule-rule1)#	**renumber**	Causes the individual rules within the message rule to be renumbered. The new sequence number assignment starts at 10 and increments by 10 for each rule.
(config)#	**hmr policy** <*name*>	Creates an HMR policy and enters the policy's configuration mode.
(config-policy-MyPolicy1)#	**rule-set** <*name*> **[**<*sequence number*>**]**	Applies a rule set to an HMR policy and optionally specifies a sequence number for the rule set within the policy. By default, sequence numbering occurs in increments of **10**, and all commands are processed in sequence. Valid sequence number range is **1** to **99999**.
(config)#	**sip hmr** <*policy name*> **[in | out]**	Applies an HMR policy to all SIP traffic and specifies whether the policy is applied to incoming or outgoing traffic.
(config-T01)#	**hmr** <*policy name*> **[in | out]**	Applies an HMR policy to a voice trunk and specifies whether the policy is applied to incoming or outgoing traffic.
(config)#	**sip proxy hmr user** <*policy name*> **[in | out]**	Applies an HMR policy to SIP proxy user traffic and specifies whether the policy is applied to incoming or outgoing traffic.

Table 7. SIP Header Manipulation Command Summary (Continued)

Prompt	Command	Description
(config)#	**sip proxy hmr server** *<policy name>* **[in | out]**	Applies an HMR policy to SIP proxy server traffic and specifies whether the policy is applied to incoming or outgoing traffic.
(config)#	**hmr set public-variable** *<variable name>* **new-value** *<pattern>*	Sets the value of a public variable. The **new-value** parameter specifies the value to assign to the variable, which can be a regular expression or text string. In addition, variable patterns can be used for manipulation, as outlined in *Variable Manipulation on page 23*.

7. Troubleshooting

After configuring SIP header manipulation in AOS, you can use the following **show** and **debug** commands to assist in troubleshooting. All of these commands are issued from the Enable mode in the CLI.

Show Commands

The variations of the **show hmr** command can be used to display statistics from traffic to which an HMR policy is applied. The output can be filtered based on policy, policy user, rule set, direction, and message type. In addition, you can sort the output by policy, policy user, rule set, direction, or message type. These commands can be used to determine the activity of an HMR policy, where the HMR policy is most used, and by whom. *Table 8* outlines the variations of the **show hmr** command.

Table 8. Show HMR Command Variations

Prompt	Command	Description
#	**show hmr**	Displays all gathered SIP header manipulation statistics.
#	**show hmr direction [in | out]**	Limits the command output to SIP HMR statistics for either inbound or outbound SIP traffic.
#	**show hmr message-type [request | response]**	Limits the command output to SIP HMR statistics for either request or response messages.
#	**show hmr policy** *<name>*	Limits the command output to SIP HMR statistics for a specified HMR policy.
#	**show hmr rule-set** *<name>*	Limits the command output to SIP HMR statistics for a specified rule set.
#	**show hmr sort [direction [in | out] | message-type [request | response] | policy** *<name>* **| user]**	Sorts SIP HMR statistics by direction, message type, policy name, or policy user.
#	**show hmr user [global | proxy-server | proxy-user |** *<user>***]**	Limits the command output to SIP HMR statistics for the SIP stack (**global**), SIP proxy servers (**proxy-server**), SIP proxy users (**proxy-user**), or a specified policy user.

Each of the **show hmr** command variations can be used multiple times within a single command. For example, you can display SIP HMR statistics for a specified policy and direction by entering the command as follows:

```
>enable
#show hmr policy MyPolicy1 direction in
```

The following example gives sample output from the **show hmr** command, which displays all HMR statistics:

```
>enable
#show hmr
Policy: MyPolicy1
Msgs Evaluated : 0
Msgs Altered : 0
User Application : Global inbound request

Rule Set          Message Rule       Action             Seq #        Count
-----------------------------------------------------------------------
Set1              Rule1              Modify Variable    10           0
                                     Modify Variable    20           0
                                     Add Header         30           3
                                     Modify Header      40           2
```

Debug Commands

Debug commands are useful in gathering statistics about SIP HMR configuration and events. The **debug hmr [traffic | configuration]** command is used to enable debug messaging for either HMR processes or HMR rules. Using the optional **traffic** parameter specifies that whenever HMR policies are applied to traffic, you receive a debug message outlining HMR message processing events. The optional **configuration** parameter specifies that whenever HMR configuration changes, you receive a debug message outlining HMR rule configuration events. If neither keyword is specified, HMR debug messages are received whenever changes in the HMR rules occur. Use the **no** form of this command to disable HMR debug messaging.

NOTE

Turning on a large amount of debug information can adversely affect the performance of your unit.

For example, to enable event messaging for HMR policies and their application to SIP traffic, enter the command as follows:

```
>enable
#debug sip stack messages
#debug hmr traffic
00:52:13.323 SIP.STACK MSG      Rx: UDP src=10.17.142.1:5060 dst=10.17.142.252:5060
00:52:13.323 SIP.STACK MSG      INVITE sip:2565550051@10.17.142.252 SIP /2.0
     00:52:13.324 SIP.STACK MSG Via: SIP/2.0/UDP
     10.17.142.1:5060;branch=z9hG4bk-2834-1-0
00:52:13.324 SIP.STACK MSG      From: 2565550052
     <sip:256555052@10.17.142.1:5060;tag=2384SIPpTag001
00:52:13.325 SIP.STACK MSG      To: 2565550051 <sip:2565550051@10.17.142.252:5060>
00:52:13.325 SIP.STACK MSG      Call-ID: 1-2384@10.17.142.1
00:52:13.326 SIP.STACK MSG      CSeq: 1 INVITE
00:52:13.327 SIP.STACK MSG      Contact: 2565550052
     <sip:2565550052@10.17.142.1:5060;transport=UDP>
00:52:13.327 SIP.STACK MSG      Max Forwards: 70
00:52:13.328 SIP.STACK MSG      Content-Type: application/sdp
00:52:13.328 SIP.STACK MSG      Content-Length: 132
```

```
00:52:13.329 SIP.STACK MSG
00:52:13.329 SIP.STACK MSG      v=0
00:52:13.329 SIP.STACK MSG      o=user1 53655765 2353687637 IN IP4 10.17.142.1
00:52:13.330 SIP.STACK MSG      s=-
00:52:13.330 SIP.STACK MSG      C=IN IP4 10.17.142.1
00:52:13.331 SIP.STACK MSG      t=0 0
00:52:13.332 SIP.STACK MSG      m=audio 10000 RTP/AVP 0
00:52:13.332 SIP.STACK MSG      a-rtpmap:0 PCMU/8000
00:52:13.332 SIP.STACK MSG
00:52:13.337 SIP.HMR PROCESS    Processing SIP message with compiled policy myPolicy
00:52:13.337 SIP.HMR PROCESS    Rule matches message, applying action rules
00:52:13.338 SIP.HMR PROCESS    Modifying SIP message headers matching from
00:52:13.338 SIP.HMR PROCESS    Header From: changed to 2565550052
     <sip:2565550052@10.17.142.1:5060>;tag=2384SIPpTag001
00:52:13.339 SIP.HMR PROCESS    Found 1 headers: Modified 1 headers using
     /(.*)(;tag.*)/
```

8. Related Online Documents and Resources

Documentation for ADTRAN AOS products is available for viewing and download directly from the ADTRAN Support Community website, available online at https://supportcommunity.adtran.com

Table 9. SIP HMR Related Online Documents and Resources

Title	Part Number	Description
Configuring the AOS Voice Loopback Account	6AOSCG0020-29	This guide describes the function and configuration of the ADTRAN AOS voice loopback account.
www.regular-expressions.info		This web site explains how regular expressions function, their use and syntax.
AOS Command Reference Guide	6000CRG0-35	This guide contains descriptions of all commands that pertain to AOS products. Content is listed in an easy to reference format.

9. Appendix A: Regular Expressions

Regular Expressions

Regular expressions, also known as **regex**, are used to describe and match specific patterns of text. Regex provide patterns that specify a set of strings. The following sections outline the very basic forms of regex, and should not be used as an all-inclusive list of regular expressions, or of regular expression operations. To fully understand how regular expressions function, their syntax, and their uses, refer to www.regular-expressions.info.

Using Regex Operations

The following section outlines some basic types of regex operations and their expressions.

NOTE

AOS CLI uses quotation marks, question marks, and spaces as native elements or delimiters in its parsing. To include any of those in a regular expression pattern as a literal string, use the hex representation instead (for example, use ***\x22*** *for quotation marks). To invoke the match of the pattern* ***/.*"Unavailable".*/****, you would use* ***/.*\x22Unavailable\x22.*/****. When specifying strings, enclosing quotes are optional unless the pattern includes spaces. Likewise, you will need to use the expression* ***%q%*** *to represent a question mark to be used as a repetition quantifier, rather than entering a* ***?*** *(question mark).*

Alternation: A Boolean OR that regex uses a vertical bar to separate alternatives, for example: entering **gray | grey** indicates that a match occurs for both **gray** and **grey**.

Grouping: Parentheses are grouping operators. For example, entering **gr(a|e)y** will also provide a match for both **gray** and **grey**. It will also capture the matching text inside the parentheses into a capture buffer, which can later be referenced by **\x**, where **x** is the number of the instance of that set of parentheses in the regex. In the previous example, since there was only one set of parentheses, you would use **\1** to reference the capture buffer.

Repetition Quantifiers: Quantifying the number of specific characters in a string can be accomplished using the characters outlined in *Table 10*.

Table 10. Repetition Quantifiers

Character	Definition
*****	Matches 0 or more of the preceding element in the string, for example: **bc*d** provides a match for **bd**, **bcd**, **bccd**, **bcccd**, etc.
+	Matches 1 or more of the preceding element in the string, for example: **bc+d** provides a match for **bcd**, **bccd**, **bcccd**, etc., but does not match **bd**.
?	Matches 0 or 1 of the preceding element in the string, for example: **colou?r** matches both **color** and **colour**. **%q%** should be used to represent the **?** used as a repetition quantifier in AOS. For example, enter **colou%q%r**.
{x}	Matches exactly **x** of the preceding element, for example: **1{2}** provides a match for **11**.
{x.}	Matches **x** or more of the preceding element, for example: **1{2},** provides a match for **11**, **111**, **1111**, etc.
{x,y}	Matches **x** through **y** of the preceding element, for example: **1{2,3}** provides a match for **11** and **111**.

Wildcards: Wildcards are used to match any single character. The only wildcard used in AOS is the . (period), which matches any single character, for example: **a.c** matches **abc**, **adc**, etc.

Character Classes: Character classes match only one of a set of several characters. *Table 11* below outlines character classes associated with regex and their uses.

Table 11. Regex Character Classes

Character	Definition
[ ]	A bracket expression matches a single character contained within the brackets. For example, **[abc]** matches **a**, **b**, or **c**.
[^]	Matches a single character not contained in the brackets. For example, **[^abc]** matches any character that is *not* **a**, **b**, or **c.**

Anchors: Anchors are used to match positions, rather than characters. *Table 12* below describes regex anchors and their uses.

Table 12. Regex Anchors

Character	Definition
^	Matches the starting position of the string.
$	Matches the ending position of the string.
\b	Matches at a word boundary. A word boundary occurs before the first character in a string if that character is not a word character, after the last character in a string if that character is not a word character, and between two characters in a string where one is a word character and the other is not. **[a-zA-Z0-9_]** are word characters.

Regex Examples

Table 13 below describes example regex and their actions.

Table 13. Regex Examples

Expression	Action
.*	Matches zero or more occurrences of any character. This will match a null string as well.
3+	Matches any string with at least one occurrence of the numeral 3.
\b300\b	Matches any string containing 300 surrounded by word boundaries.
\b300$	Matches any string ending in 300 with a word boundary before it.
^300\b.*	Matches any string that begins with 300 followed by a word boundary.
^300\b.*301	Matches any string that starts with 300, followed by a word boundary, and ends with 301.
^300$	Matches any string that contains only 300.

10. Appendix B

Matching Regular Expressions in SIP Messages with Multiples of the Same Header

Some SIP messages include multiple headers of the same type. For example, you might have a SIP message with multiple Contact, Via, Route, Record-Route, Allow, etc. headers. The format for expressing these headers within the SIP packet may differ. For example, these headers can be expressed in multiple header lines within the packet, or multiple header values can be included on a single header line using comma separated values (CSVs).

If you are using a regex to match SIP headers that have multiple values defined, you must take into account the notation method of those headers when configuring your match criteria. In the example below, there are three contact headers for Alice in Atlanta.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via:SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>
Contact: <sip:alice@pc34.atlanta.com>
Contact: <sip:alice@pc35.atlanta.com>
```

When multiple headers of the same type are expressed in multiple lines (as they are in this example), the first contact header (**<sip:alice@pc33.atlanta.com>**) is in the **first** position and the third contact header (**<sip:alice@pc35.atlanta.com>**) is in the **last** position. When you specify the position of the matching criteria for HMR, whether you are adding, removing, or modifying the header, you would enter the command as follows to match based on the contact header in the **last** position:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#message-rule Rule1 message-type any 1
(config-msg-rule-Rule1)#modify header position last
```

Multiple SIP headers of the same type are not always expressed in multiple header lines. For example, you could see multiple headers of the same type expressed in the same line, separated by commas. The example below uses the same information as the previous example, except that all the contact headers are in a single line.

```
INVITE sip:bob@biloxi.com SIP/2.0
Via:SIP/2.0/UDP pc33.atlanta.com;branch=z9hG4bKnashds8
To: Bob <sip:bob@biloxi.com>
From: Alice <sip:alice@atlanta.com>;tag=1928301774
Call-ID: a84b4c76e66710
CSeq: 314159 INVITE
Max-Forwards: 70
Contact: <sip:alice@pc33.atlanta.com>, <sip:alice@pc34.atlanta.com>,
     <sip:alice@pc35.atlanta.com>
```

When multiple headers of the same type are expressed in a single line (as they are in this example), the first contact header (**<sip:alice@pc33.atlanta.com>**) is in the **first** position and the third contact header (**<sip:alice@pc35.atlanta.com>**) is in the **last** position. It is important to understand that for matching purposes, each value separated by the comma is a different contact header. When you specify the position of the matching criteria for HMR, whether you are adding, removing, or modifying the header, you would enter the command as follows to match based on the contact header in the **last** position:

```
(config)#hmr rule-set Set1
(config-rule-set-Set1)#message-rule Rule1 message-type any 1
(config-msg-rule-Rule1)#modify header position last
```

Because header information can be expressed in a single line, multiple lines, or a combination of the two, you must be careful when specifying header positions for matching purposes. In addition, you might need to use the **first-match** keyword of the **position** parameter to specify a header that is not in the first or last header position.

11. Warranty and Contact Information

Warranty

Warranty information can be found at:

www.adtran.com/warranty.

Contact Information

For all customer support inquiries, please contact ADTRAN Customer Care:

Table 14.

Contact	Support	Contact Information
Customer Care	From within the U.S. From outside the U.S. **Technical Support:** ■ Web: **Training:** ■ Email: ■ Web:	1-888-4ADTRAN (1-888-423-8726) + 1 (256) 963-8716 www.adtran.com/support training@adtran.com www.adtran.com/training www.adtranuniversity.com
Sales	Pricing and Availability	1-800-827-0807