How To Guide

Generic Header Manipulation
&
Regular Expressions

For the Ingate SIParators® and Firewalls using software release 4.10.x or later.
Updated to show features available from release 5.0.4

15/04/2019

Revision History:

<table>
<thead>
<tr>
<th>Revision</th>
<th>Date</th>
<th>Author</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2010-11-03</td>
<td>Scott Beer</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>2010-11-16</td>
<td>SB and KES</td>
<td>First Release</td>
</tr>
<tr>
<td>3</td>
<td>2012-05-23</td>
<td>Scott Beer</td>
<td>4.10 updates</td>
</tr>
<tr>
<td>4</td>
<td>2012-05-23</td>
<td>KES</td>
<td>Front page only</td>
</tr>
<tr>
<td>5</td>
<td>2015-07-10</td>
<td>RN</td>
<td>Major rewrite and 5.0.4 updates</td>
</tr>
<tr>
<td>6</td>
<td>2015-09-24</td>
<td>KES</td>
<td>Renamed to reflect RegExp also</td>
</tr>
<tr>
<td>7</td>
<td>2016-01-19</td>
<td>PD</td>
<td>Minor content updates</td>
</tr>
<tr>
<td>8</td>
<td>2016-02-23</td>
<td>PD / AG</td>
<td>Add ${{syntax}}</td>
</tr>
<tr>
<td>9</td>
<td>2016-11-21</td>
<td>PD</td>
<td>Minor updates</td>
</tr>
<tr>
<td>10</td>
<td>2016-12-19</td>
<td>PD</td>
<td>Errata bug catch, minor fixes</td>
</tr>
<tr>
<td>11</td>
<td>2017-04-16</td>
<td>PD / AG</td>
<td>Add 6.0.x GHM variable</td>
</tr>
<tr>
<td>12</td>
<td>2017-10-26</td>
<td>PD</td>
<td>Modifications to CVS/CHM</td>
</tr>
<tr>
<td>13</td>
<td>2018-05-28</td>
<td>PD</td>
<td>New regexps, some rewrites</td>
</tr>
<tr>
<td>14</td>
<td>2019-03-28</td>
<td>PD</td>
<td>Minor edits</td>
</tr>
</tbody>
</table>
Table of Contents

Errata .................................................................................................................. 4

1 Introduction .................................................................................................... 5
  1.1 Example of Generic Header Manipulation ............................................... 5

2 Generic Header Manipulation (GHM) – Why do we need GHM? ................. 6
  2.1 Where to configure GHM ....................................................................... 6
  2.2 Incoming messages ............................................................................... 9
  2.3 Outgoing messages ............................................................................. 10

3 Regular Expressions ...................................................................................... 12
  3.1 Industry-standard regular expressions notation ................................... 12
  3.2 Routing of calls using the Dial Plan and the SIP trunk Page ..................... 13
  3.3 Examples of Basic Regular Expressions ............................................. 17
  3.4 183 to 180 conversion ........................................................................ 18
  3.5 Do not REGISTER to trunk server(s) .................................................... 18
  3.6 Do not automatically monitor trunk server(s) with SIP OPTIONS ............. 18
  3.7 B2bua with media via the main dial plan .......................................... 18
  3.8 Explicitly stating transport .................................................................. 19
  3.9 Specifying Escape Characters (dial string) for e.g. Telia SIP trunk .......... 19
  3.10 Explicitly handling only specific METHODS ...................................... 19
  3.11 Additional information ..................................................................... 20

4 Conditional Regular Expressions (CRE) ...................................................... 21
  4.1 Example CRE - Sweden Number Reduction ....................................... 21
  4.2 Example CRE - USA Number Reduction .......................................... 21
  4.3 Generic Number Reduction to + prefix for phone numbers 8-12 digits long ................................................................. 22

5 Variable Substitution ................................................................................. 23
  5.1 Port and Password ............................................................................ 24
  5.2 Examples ......................................................................................... 26

6 Generic Header Manipulation (GHM) .......................................................... 27
  6.1 Adding or Replacing a Header ............................................................ 27
  6.2 Removing a header ........................................................................... 28
  6.3 Generic Header Manipulation (GHM) for Responses ......................... 28
  6.4 Multiple Occurrences of Headers ..................................................... 29
  6.5 Variable substitution ....................................................................... 29

7 Conditional Variable substitution (CVS) .................................................... 31
  7.1 Conditional Header Manipulation (CHM) .......................................... 31
  7.2 URI Parameter Chaining .................................................................... 34

8 Keyword / Grammar Summary – CVS, CHM and CRE ............................... 36
  8.1 Tests ............................................................................................. 36
  8.2 Results ........................................................................................... 36
  8.3 RegExp ......................................................................................... 36

File name: How_To_use_Generic_Header_Manipulation.pdf
Terminology used in this document:

<table>
<thead>
<tr>
<th>Abbreviation</th>
<th>Term</th>
</tr>
</thead>
<tbody>
<tr>
<td>GHM</td>
<td>Generic Header Manipulation</td>
</tr>
<tr>
<td>CRE</td>
<td>Conditional Regular Expressions</td>
</tr>
<tr>
<td>CVS</td>
<td>Conditional Variable Substitution</td>
</tr>
<tr>
<td>CHM</td>
<td>Conditional Header Manipulation</td>
</tr>
<tr>
<td>B2BUA</td>
<td>Back to Back User Agent</td>
</tr>
<tr>
<td>ITSP</td>
<td>Internet Telephony Service Provider</td>
</tr>
</tbody>
</table>
Errata:

Note: In firmware versions <= 5.0.11 you cannot use $1?From=$(......$1...) – i.e. regexp capture result groups (on the trunk page i.e. where $1 is before and after the “?”), and $REGMATCH expressions which also contain their own $1 capture groups. The workaround is to use $0?From=$(....$1...).

Expressions such as the following will fail to evaluate:

\[\text{sip:}\$0\text{?From}=@192.168.1.1?From=%3Csip%3a$\{\text{REGMATCH} \_^{001}(\[0-9]\{10\}\})$\_\text{REGMOD}_+1\$1\_\text{REGELSE} \_^{1}(\[0-9]\{10\}\})$\_\text{REGMOD}_+1\$1\_\text{REGEND}.from.user)%40\{\text{from.host}\}\{\{\text{from.uriparams}\}\} %3E\{\text{from.params}\}\]

A workaround is thus:

\[\text{sip:}\$0?From=@192.168.1.1?From=%3Csip%3a$\{\text{REGMATCH} \_^{001}(\[0-9]\{10\}\})$\_\text{REGMOD}_+1\$1\_\text{REGELSE} \_^{1}(\[0-9]\{10\}\})$\_\text{REGMOD}_+1\$1\_\text{REGEND}.from.user)%40\{\text{from.host}\}\{\{\text{from.uriparams}\}\} %3E\{\text{from.params}\}\]

$0$ evaluates to the whole user portion of the RURI on the trunk page.

Or not to use a capture group on the trunk page e.g.

<table>
<thead>
<tr>
<th>Incoming Trunk Match</th>
<th>Forward to</th>
</tr>
</thead>
<tbody>
<tr>
<td>(.* )</td>
<td>$1</td>
</tr>
</tbody>
</table>

While instead use:

<table>
<thead>
<tr>
<th>Incoming Trunk Match</th>
<th>Forward to</th>
</tr>
</thead>
<tbody>
<tr>
<td>.*</td>
<td>$0</td>
</tr>
</tbody>
</table>

If a fix is necessary e.g. you must forward trunk captures to a host which differs from the PBX configured on the trunk page, write to support@ingate.com for a patch for your current firmware version <= 5.0.11, or upgrade your firmware > 5.0.11
1 Introduction
This document describes how to use Generic Header Manipulation (GHM) in an Ingate SIParator/Firewall. With the GHM feature it is possible to Add, Change or Remove any SIP Header of incoming or outgoing SIP messages, both Requests and Responses. The purpose of this feature is to enhance the interoperability between different vendor equipment, as many IP-PBX, Service Providers, and overall SIP devices require different usage of various SIP Headers. Different vendors interpret the SIP standard in a non-conformant way and by using GHM can SIP protocol variances be resolved.

Rules are configured in the Dial Plan and SIP Trunk Page GUIs. The rules are configured in the same fields as regular expressions are written. With regular expressions, it is possible to match substrings and variables from SIP messages. When forwarding SIP messages, they can be rewritten according to your wishes. By variable substitutions it is also possible to take information from one header and put it in another header.

Note: Text written in a Monospace Font signifies regular expression code i.e. GHM, CHM, CRE, CVS or processing result.

Note: Result samples are not intended to be protocol-accurate or compliant; they are intended for demonstration only and reduced for simplicity.

1.1 Example of Generic Header Manipulation

For example, when a Service Provider requires the use of a P-Asserted-Identity Header and the IP-PBX is unable to provide this SIP Header, the Ingate can add in a P-Asserted-Identity Header. As seen here:

```
INVITE sip:61355512128209.216.177.59:5060 SIP/2.0
Via: SIP/2.0/UDP 209.249.3.100:5060;branch=Z9hG4bK9624349
To: sip:61355512128209.249.3.56:5060
From: <sip:50355511110209.249.3.100>;tag=3462103187-665679
Supported: timer, 100rel
Call-ID: 2165939-3462103187-665672@NXT02.broadvox.net
CSeq: 1 INVITE
Allow: INVITE, BYE, OPTIONS, CANCEL, ACK, REGISTER, PRACK, UPDATE
Max-Forwards: 69
Session-Expires: 3600;refresher=uac
Contact: sip:50355511110209.249.3.100:5060
Content-Type: application/sdp
Content-Length: 249
```

The additional header will be added into the SIP message when it is forwarded through the Ingate unit.

```
INVITE sip:61355512128209.216.177.59:5060 SIP/2.0
Via: SIP/2.0/UDP 209.249.3.100:5060;branch=Z9hG4bK9624349
To: sip:61355512128209.249.3.56:5060
From: <sip:50355511110209.249.3.100>;tag=3462103187-665679
Supported: timer, 100rel
Call-ID: 2165939-3462103187-665672@NXT02.broadvox.net
CSeq: 1 INVITE
Allow: INVITE, BYE, OPTIONS, CANCEL, ACK, REGISTER, PRACK, UPDATE
Max-Forwards: 69
Session-Expires: 3600;refresher=uac
Contact: sip:50355511110209.249.3.100:5060
P-Asserted-Identity:sip:5035551111@64.156.174.74:5060
Content-Type: application/sdp
Content-Length: 249
```
Generic Header Manipulation (GHM) – Why do we need GHM?

The GHM feature allows arbitrary manipulation of SIP headers when the SIP messages are transmitted through the Ingate Firewall/SIParator. It is possible to Add, Change or Remove any SIP Header of incoming or outgoing SIP messages, both Requests and Responses. In addition, the GHM feature can use regular expressions to modify existing SIP headers, add SIP headers, and to match SIP headers to store variable information.

The following list summarizes the supported GHM actions for a SIP header:
- Add a header to the message, if the header doesn’t exist
- Replace header content
- Pass a header unchanged
- Remove a header
- Replace an instance of a header

All listed actions can be done conditionally – i.e. check for a matching condition, and then act. Header manipulation for requests are performed after routing of calls, while responses are modified prior to routing.

2.1 Where to configure GHM

Regular expressions for configuring matches from SIP message are configured in the below listed locations within the Ingate Firewall/SIParator (marked with blue in the figures below).

Rules for regular expression substitution and header manipulation of SIP messages are configured in the Forward To type of fields in the below listed locations within the Ingate Firewall/SIParator (marked with red in the figures below).

### Table: Regular Expression match vs. Header manipulation/Regular Expression Substitution

<table>
<thead>
<tr>
<th>Regular Expression match</th>
<th>Header manipulation/Regular Expression Substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>SIP Traffic &gt; Dial Plan</td>
<td>SIP Traffic &gt; Dial Plan</td>
</tr>
<tr>
<td>Matching From Header – Reg Expr</td>
<td>Forward To – Reg Expr</td>
</tr>
<tr>
<td>Matching Request-URI – Reg Expr</td>
<td>SIP Traffic &gt; Routing</td>
</tr>
<tr>
<td></td>
<td>User Routing – Forward To</td>
</tr>
<tr>
<td></td>
<td>Static Registrations – Forward To</td>
</tr>
<tr>
<td>SIP Trunks &gt; Trunk n page</td>
<td>SIP Trunks &gt; Trunk n page</td>
</tr>
<tr>
<td>Main Trunk Line – Incoming Trunk Match</td>
<td>Main Trunk Line – Username and Forward</td>
</tr>
<tr>
<td>PBX Lines – From PBX Number/User and Incoming Trunk Match</td>
<td>PBX Lines – Username and Forward To PBX Account</td>
</tr>
<tr>
<td>SIP Lines – From SIP Number/User and Incoming Trunk Match</td>
<td>SIP Lines – Username and Forward To SIP Account</td>
</tr>
</tbody>
</table>

The purpose of the Matching From Header is to provide source SIP URI matching, as well as source Transport and Network Address. In the Regular Expression, you create criteria for the From Header
SIP URI of the SIP messages. This is used when matching requests in the Dial Plan. For a request to match, all criteria must be fulfilled.

The purpose of the Matching Request URI is to match on incoming Request URI Header of the SIP messages being sent to the Ingate. Typically, the “domain” portion of the URI is the Ingate IP Address or FQDN. In the Regular Expression, you create criteria for the Request-URI of the SIP messages. This is used when matching requests in the Dial Plan. For a request to match, all criteria must be fulfilled.

Port and Transport can be used but it only matches if there's both a port and a transport parameter contained in the SIP URI

The purpose of the Forward To attribute of the Dial Plan is to define where to send the SIP traffic. A specific destination SIP URI address is defined to forward the call to. Here you may enter Regular Expressions for the Dial Plan, used to define where and how the firewall should forward the request using the Dial Plan. Here you may also enter rules for GHM and variable substitution.

SIP Traffic – Routing

With Static Registrations you can specify that calls to a certain user address should also be redirected to another address, or that calls to a non-person user name (like support@company.com) should be redirected to one or more other addresses. Even if a call should be forwarded, the SIParator will try to put it through to the original addressee.

The User Routing table makes it possible to allow advanced routing options to be enabled per user. You may enter aliases that are used to match incoming request to a specific user. Additionally you
may define that the request should be forwarded to other users, and also set up connections to voice mail.
In respective Forward to field you may use Variable substitution and GHM, but not regular expressions.

**SIP Trunk – Trunk 1-n**

**From PBX/SIP Number/User** - Can contain a number/name or regular expression. For outgoing calls, this field is matched to the calling user's user name in the From SIP URI (often a number but can also be a name). If there is a match, this line is selected for usage for the outgoing call.

**User Name** - The SIP user name to use in the From SIP URI for outgoing calls and registrations. This is usually the telephone number of the ITSP SIP account and usually the number displayed as caller ID on the PSTN. In this field you can use the result of a sub-expression from a match in a regular expression defined in the "From Number/User" field on the same row.
2.2 Incoming messages

The SIP Trunk page is designed to connect IP-PBXs and other SIP endpoints (phones) to an ITSP's (Internet Telephony Service Provider's) SIP Trunking service using a B2BUA (Back-To-Back User Agent) with one interface to the ITSP and the other to the IP-PBX and SIP phones.

For incoming SIP messages the rules for header manipulation of SIP messages are configured in the GUI for the SIP Trunk Page, in the columns of Incoming Trunk Match and Forward to for PBX Lines, SIP Lines, and for Main Trunk Line.

For every incoming SIP message the rows in Incoming Trunk Match, for PBX Lines and SIP Lines, are gone through row for row until match. In case of no match the values from Main Trunk Line are used.

When there is a match in Incoming Trunk Match field, the call will be forwarded according to the values in respective Forward to field at the same row. For PBX Lines, this is the number an incoming call will be forwarded to on the PBX. For SIP Lines, this is the SIP address (complete SIP URI or a SIP user name/number on this unit's SIP Server) an incoming call will be forwarded to.

Regular expressions can be used in the Incoming Trunk Match field to catch information from the R-URI to be used in the Forward to field, where also variable substitution and GHM can be made.
An example of a regular expression is:

<table>
<thead>
<tr>
<th>Incoming Trunk Match</th>
<th>Forward to</th>
</tr>
</thead>
<tbody>
<tr>
<td>+1306(7707[0-9]{2})</td>
<td>0$1; user=phone</td>
</tr>
</tbody>
</table>

This will result in an incoming call to +1306770713, forwards to the PBX number 0770713. Here you may also add URI parameters such as ",;user=phone" at the end of the domain of a complete URI.

### 2.3 Outgoing messages

Outgoing calls are processed through the Dial Plan. The Dial Plan table is searched line by line from the top for a match from the PBX of the dialed number where after it is forwarded to the SIP Trunk page.

At the SIP Trunk Page, the caller’s number (the user part of the From header) will - row by row, from top to bottom - be checked against the numbers or regular expressions entered in the column From.
**PBX/SIP Number/User.** A match will cause the trunk account under **User Name** to be used for the outgoing call.

The results of subexpressions from a match in the column **From PBX/SIP Number/User** can be used in the columns **Display Name**, **User Name** and **Identity**.

To understand how GHM works it is first necessary to understand how regular expressions are used, see next chapter.
3 Regular Expressions
Within the SIP Protocol (RFC 3261), a SIP URI identifies a communications resource. Like all URIs, SIP URIs may be placed in web pages, email messages, or printed literature. They contain sufficient information to initiate and maintain a communication session with the resource. In its simplest form a SIP URI looks like sip:user@host, where the "user" is the identifier of a particular client resource at the host being addressed. The term "host" in this context frequently refers to a domain, a network location.

When using Regular Expressions to define a SIP URI, these expressions allow for the customization and flexibility to define SIP URIs as variables. Regular expressions can increase the capabilities of the Ingate Dial Plan and SIP trunk pages as it further increases the definitions of the SIP URIs used to route calls. The Regular Expression can be used to define both the user and host portion of the SIP URI.

3.1 Industry-standard regular expressions notation
Regular Expression flavor used in Ingate SIParator/Firewall is “POSIX Extended Regular Expressions (ERE)”.

Note: Character matches are case sensitive.

<table>
<thead>
<tr>
<th>Operator</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>[]</td>
<td>A bracket expression. Matches any single character that is contained within the brackets. For example: [abc] Matches any single character in the set a, b, or c. [a-z] Matches any single character in the range a-z but not A-Z [1-8] Matches any single character in the range 1 to 8. [369] Matches any single character in the set 3, or 6, or 9.</td>
</tr>
<tr>
<td>[^ ]</td>
<td>Matches any single character that is not contained within the brackets. For example: [^abc] Matches any single character not in the set a, b, or c.</td>
</tr>
<tr>
<td>.</td>
<td>Matches any single character.</td>
</tr>
<tr>
<td>,</td>
<td>Matches the minimum specified characters or more.</td>
</tr>
<tr>
<td>[0-9]</td>
<td>Matches any decimal digit.</td>
</tr>
<tr>
<td>[^0-9]</td>
<td>Matches any non-digit.</td>
</tr>
<tr>
<td>\s</td>
<td>Matches any whitespace character.</td>
</tr>
<tr>
<td>\S</td>
<td>Matches any non-whitespace character.</td>
</tr>
<tr>
<td>\w</td>
<td>Matches any word (alphanumeric) character.</td>
</tr>
<tr>
<td>\W</td>
<td>Matches any non-word (alphanumeric) character.</td>
</tr>
<tr>
<td>(abc)</td>
<td>Defines a marked sub-expression. For example: (abc) Matches the sequence abc and stores it as a variable which may be used in later expressions. Also used for grouping.</td>
</tr>
<tr>
<td>$1</td>
<td>The $ symbol is used to recall expressions that have been stored within ()-variables which are numbered according to the capture hierarchy. $1 refers to the first variable stored, and $2 refers to the second variable stored, etc.</td>
</tr>
<tr>
<td>a</td>
<td>b</td>
</tr>
<tr>
<td>+</td>
<td>Matches the preceding expression one or more times.</td>
</tr>
<tr>
<td>?</td>
<td>Makes preceding expression optional; if the preceding is inside () brackets, e.g. (345)? then that 345 is optional.</td>
</tr>
<tr>
<td>*</td>
<td>Matches the null string or any number of repetitions of the preceding expression.</td>
</tr>
<tr>
<td>{m}</td>
<td>Matches exactly m repetitions of the one-character expression.</td>
</tr>
<tr>
<td>{m,n}</td>
<td>Matches between m and n repetitions of the preceding expression, inclusive.</td>
</tr>
<tr>
<td>{m,}</td>
<td>Matches m or more repetitions of the preceding expression.</td>
</tr>
<tr>
<td>^</td>
<td>Matches the starting position within the string.</td>
</tr>
<tr>
<td>----</td>
<td>------------------------------------------------</td>
</tr>
<tr>
<td>$</td>
<td>Matches the ending position of the string.</td>
</tr>
</tbody>
</table>

### 3.1.1 Examples

<table>
<thead>
<tr>
<th>Pattern</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>(.*innen)</td>
<td>Stores any amount of characters in $1</td>
</tr>
<tr>
<td>sip:(.<em>)@(.</em>)</td>
<td>Stores user in $1 and host in $2 if applied to sip:user@host</td>
</tr>
</tbody>
</table>

Sub-expressions are numbered in the order of their starting parenthesis and referred to as $number. In the expression sip:(.*)@(.*) which matches any Request-URI like sip:user@ingate.com, there are two referable sub-expressions: user, which is referred to as $1, and ingate.com, which is referred to as $2.

Sub-expression can also be nested, as in the expression (sip:(.*))@ingate.com, which matches any Request-URI like sip:user@ingate.com, there are two referable sub-expressions: sip:user, which is referred to as $1, and user, which is referred to as $2.

### 3.1.2 Escape sequence for special characters:

Meta characters are characters with a special meaning in Regular Expressions. There are 12 characters with special meanings: “\”, “^”, “$”, “.”, “|”, “?” , “*”, “+”, “(”, “)”, “[”, “]”. If you want to use any of these characters as a literal in a Regular Expression, you need to escape them with a backslash. For example “+”-sign will be \+ and +46701234567 will be \+46701234567.

### 3.2 Routing of calls using the Dial Plan and the SIP trunk Page

The dial plan and the SIP Trunk pages are used to instruct the Ingate SIP Proxy or B2BUA about how to route calls. Who to accept calls from, match on where calls are going to, and where to send calls. Adding regular expressions into the dial plan and the SIP trunk page allows you to specify a range of numbers, range of domains, or other set of specific digits.

Regular expressions are a flexible way of delivering patterns that match a unique set of criteria. For example, if you specify the regular expression [0-9][7,] Ingate Firewall/SIParator recognizes any digits from zero to nine, repeated seven or more times, in other words, a telephone number.

### 3.2.1 Regular Expressions in the Matching From Header

The purpose of the Matching From Header is to provide source SIP URI matching, that is matching the FROM: field or the origin of a call.
<table>
<thead>
<tr>
<th>SIP URI Example Description</th>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-digit number @ Any Domain</td>
<td>sip:[0-9]{7}@.*</td>
</tr>
<tr>
<td>7-digit number @ IP Address</td>
<td>sip:[0-9]{7}@12.34.56.78</td>
</tr>
<tr>
<td>7-digit number @ Domain</td>
<td>sip:[0-9]{7}@sip_domain.com</td>
</tr>
<tr>
<td>7-digit number @ (anything hereafter doesn’t matter)</td>
<td>sip:[0-9]{7}(@.*)</td>
</tr>
<tr>
<td>North American long-distance number @</td>
<td>sip:1[2-9]{0-9}{9}@</td>
</tr>
<tr>
<td>North American Toll-free number: 1+800+7 digits @</td>
<td>sip:18(00</td>
</tr>
<tr>
<td>Local 7-digit number, beginning with optional 9 @</td>
<td>sip:9?[0-9]{7}@</td>
</tr>
<tr>
<td>4-digit number (as an extension) starting with 5 @</td>
<td>sip:5[0-9]{3}@</td>
</tr>
<tr>
<td>4-digit number not starting with 36 @</td>
<td>sip:(?36)[0-9]{4}@</td>
</tr>
<tr>
<td>Any Username @ Any Domain</td>
<td>sip:.<em>@.</em></td>
</tr>
<tr>
<td>Any Username @ IP Address</td>
<td>sip:.*@12.34.56.78</td>
</tr>
<tr>
<td>Any Username @ Domain</td>
<td>sip:.*@sip_domain.com</td>
</tr>
<tr>
<td>UK 7-digit numbers within the London area codes 0207 and 0208@</td>
<td>sip:020[78]{0-9}{7}@</td>
</tr>
<tr>
<td>UK 7-digit number with 0845 Lo-call prefix @</td>
<td>sip:0845[0-9]{7}@</td>
</tr>
<tr>
<td>UK 6 to 7-digit number with 0845 Lo-call prefix @</td>
<td>sip:0845[0-9]{6,7}@</td>
</tr>
<tr>
<td>UK 6-digit number with 0845 Lo-call prefix @</td>
<td>sip:0845[0-9]{6}@</td>
</tr>
<tr>
<td>UK 7-digit number with 0870 National prefix @</td>
<td>sip:0870[0-9]{7}@</td>
</tr>
<tr>
<td>UK 6 to 7-digit number with 0870 National prefix @</td>
<td>sip:0870[0-9]{6,7}@</td>
</tr>
<tr>
<td>UK 6-digit number with 0870 National prefix @</td>
<td>sip:0870[0-9]{6}@</td>
</tr>
<tr>
<td>UK 7-digit number with 0845 Lo-call or 0870 National prefix @</td>
<td>sip:08[74]{05}[0-9]{7}@</td>
</tr>
<tr>
<td>Note: the first will also match 0875 or 0840, the second won’t</td>
<td>sip:(0870</td>
</tr>
<tr>
<td>UK Any 9 to 10-digit numbers prefixed with optional OO and then mandatory 44</td>
<td>sip:00?44[0-9]{9,10}@</td>
</tr>
<tr>
<td>UK 7-digit Liverpool 151 numbers prefixed with O</td>
<td>sip:0151[0-9]{7}@</td>
</tr>
</tbody>
</table>

### 3.2.2 Regular Expressions in the Matching Request URI

The purpose of the Matching Request-URI is to match a Request URI Header of the SIP messages being sent through the Ingate and determine where it wants to go. Typically the "domain" portion of the URI is the Ingate IP Address or FQDN. In the Regular Expression, you create criteria for the Request-URI of the SIP messages. This is used to match requests in the Dial Plan. For a request to match, all criteria must be fulfilled. Port and Transport can be used but will only match if there are both port and transport parameters contained in incoming the SIP URI.
<table>
<thead>
<tr>
<th>SIP URI Example Description</th>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>7-digit number @ Any Domain</td>
<td>sip:([0-9]{7})@.*</td>
</tr>
<tr>
<td>7-digit number @ IP Address</td>
<td>sip:([0-9]{7})@12.34.56.78</td>
</tr>
<tr>
<td>7-digit number @ Domain</td>
<td>sip:([0-9]{7})@sip_domain.com</td>
</tr>
<tr>
<td>7-digit number @ (anything hereafter doesn’t matter)</td>
<td>sip:([0-9]{7})@(.*)</td>
</tr>
</tbody>
</table>

Emergency numbers 112 or 999 @

Warning: not all SIP providers have access to emergency service numbers.

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:112</td>
</tr>
</tbody>
</table>

North American long-distance number @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:1[2-9][0-9]{9}@</td>
</tr>
</tbody>
</table>

North American Toll-free number: 1+800+7 digits @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:18(00</td>
</tr>
</tbody>
</table>

Local 7-digit number, beginning with optional 9 @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:9?[0-9]{7}@</td>
</tr>
</tbody>
</table>

4-digit number (as an extension) starting with 5 @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:5[0-9]{3}@</td>
</tr>
</tbody>
</table>

4-digit number not starting with 36 @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip: (?36) [0-9]{4}@</td>
</tr>
</tbody>
</table>

Remove Prefix "1613" on any Username @

Anything $1 is (.*) i.e. 1613 is matched but not stored. If the number doesn't begin "1613", there will be no match.

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:1613(.<em>?)@(.</em>)</td>
</tr>
</tbody>
</table>

Remove Prefix "1613" on 11-digit number @

Anything and pass only 7-digits

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:1613([0-9]{7})@(.*)</td>
</tr>
</tbody>
</table>

Remove optional Prefix "+" on any Username @

Note: some characters need to have an escape sequence, among them the "+" character.

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:+?(.*)@</td>
</tr>
</tbody>
</table>

Any Username @ Any Domain

Any Username @ IP Address

Any Username @ Domain

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:(.<em>)@.</em></td>
</tr>
<tr>
<td>sip:(.*)@12.34.56.78</td>
</tr>
<tr>
<td>sip:(.*)@sip_domain.com</td>
</tr>
</tbody>
</table>

Any Username @ Any Domain with Port

Any Username @ IP Address with Port

Any Username @ Domain with Port

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:(.<em>)@.</em>:5060</td>
</tr>
<tr>
<td>sip:(.*)@12.34.56.78:5060</td>
</tr>
<tr>
<td>sip:(.*)@sip_domain.com:5060</td>
</tr>
</tbody>
</table>

Any Username @ Any Domain with Port and Transport

Any Username @ IP Address with Port and Transport

Any Username @ Domain with Port and Transport

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip: (?36) @.*:5060;transport=UDP</td>
</tr>
<tr>
<td>sip: (?36) @12.34.56.78:5060;transport=UDP</td>
</tr>
<tr>
<td>sip: (?36) @sip_domain.com:5060;transport=UDP</td>
</tr>
</tbody>
</table>

UK 7-digit numbers within the London area codes 0207 and 0208 @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:020[78][0-9]{7}@</td>
</tr>
</tbody>
</table>

UK 7-digit number with 0845 Lo-call prefix @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:0845[0-9]{7}@</td>
</tr>
</tbody>
</table>

UK 6 to 7-digit number with 0845 Lo-call prefix @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:0845[0-9]{6,7}@</td>
</tr>
</tbody>
</table>

UK 6-digit number with 0845 Lo-call prefix @

<table>
<thead>
<tr>
<th>Equivalent Regular Expression</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:0845[0-9]{6}@</td>
</tr>
</tbody>
</table>
UK 7-digit number with 0870 National prefix @
UK 6 to 7-digit number with 0870 National prefix @
UK 6-digit number with 0870 National prefix @

<table>
<thead>
<tr>
<th>Pattern</th>
<th>URI</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:0870[0-9]{7}@</td>
<td>sip:0870[0-9]{6,7}@</td>
</tr>
<tr>
<td>sip:08[74][05][0-9]{7}@</td>
<td>sip:(0870</td>
</tr>
<tr>
<td>sip:00?44[0-9]{9,10}@</td>
<td>sip:0151[0-9]{7}@</td>
</tr>
<tr>
<td>sip:(((00)?353)?0?[0-9]{1,2})?([0-9]{7})@.*</td>
<td></td>
</tr>
</tbody>
</table>

Note: that in all of the above expressions, there is no match if the RURI isn't pre-pended with “sip:”, i.e. requests beginning with “tel:” will not match. Also, while sip: is matched, it isn't stored in any of the above examples.

3.2.3 Regular Expressions in the Forward To
The Forward To attribute of the Dial Plan defines where and how to send SIP traffic. A specific destination SIP URI address is defined to forward the call to.
Here you may enter Regular Expressions for the Dial Plan, used to define where and how the firewall should forward the request using the Dial Plan. A Regular Expression refers to RegExp sub-expressions on the corresponding row in the Matching Request-URI table. Sub-expressions are numbered in the order of their starting parenthesis and referred to as $number.
You may define lines in the Dial Plan that lack a Forward to definition. This is useful if you for example are forwarding by ENUM.
### SIP URI Example Description | Equivalent Regular Expression
---|---
Fixed number: 911 @ IP Address | sip:911@12.34.56.78
Fixed number: 911 @ Domain | sip:911@sip_domain.com
Fixed 7-digit number @ IP Address | sip:9630933@12.34.56.78
North American long-distance number @ Domain | sip:16139630933@sip_domain.com
North American Toll-free number: 1+800+7 digits @ Domain | sip:18668090002@sip_domain.com
Use Stored Variable @ IP Address | sip:$1@12.34.56.78
Use Stored Variable @ Domain | sip:$1@sip_domain.com
Use Stored Variable @ Domain with Port and Transport | sip:$1@sip_domain.com:5060;transport=UDP
Use To Header in Request URI @ Domain | sip:$(to.user)@sip_domain.com
Use To Header in Request URI and To Host in Domain and send to specified address | sip:$(to.user)@$(to.host);maddr=12.34.56.78
Add +1 in front of To Header in Request URI @ Domain | sip:+1$(to.user)@sip_domain.com

**Note:** The table includes examples of variable substitution, e.g. $(to.user), see chapter 5 Variable Substitution for explanation.

### 3.3 Examples of Basic Regular Expressions

Here are some basic examples of some standard Regular Expressions to be used in the Forward To columns either of the Dial Plan or the SIP Trunk page (without the use of the Generic Header Manipulation). The examples assume that there is a match done resulting in that $1 contains the user part of the Request-URI.

<table>
<thead>
<tr>
<th>Using Proxy</th>
<th>Adding Port parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:$1@192.168.1.1</td>
<td>sip:$1@192.168.1.1:5060</td>
</tr>
<tr>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a></td>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a>:5060</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adding B2BUA</th>
<th>Adding Transport parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:$1@192.168.1.1;b2bua</td>
<td>sip:$1@192.168.1.1;transport=UDP</td>
</tr>
<tr>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a>;b2bua</td>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a>;transport=UDP</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adding Prefix</th>
<th>Adding From Header parameter</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:+$1@192.168.1.1</td>
<td>sip:+$1@192.168.1.1;from=&quot;sip:$(from.user)@1.2.3.4&quot;</td>
</tr>
<tr>
<td>sip:+$<a href="mailto:1@example.com">1@example.com</a></td>
<td>sip:+$<a href="mailto:1@example.com">1@example.com</a>;from=&quot;sip:$(from.user)@1.2.3.4&quot;</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Adding Any Combination (This is an example of adding Transport, Port, B2BUA and From Header)</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:$1@192.168.1.1:5060;transport=TCP;b2bua;from=&quot;sip:$(from.user)@1.2.3.4&quot;</td>
</tr>
<tr>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a>:5060;transport=TCP;b2bua;from=&quot;sip:$(from.user)@1.2.3.4&quot;</td>
</tr>
</tbody>
</table>

**Note:** the Adding From Header parameter includes an example of Variable Substitution. See chapter 5 Variable Substitution for clarification.
**Note:** the *From Header* is the only header that can be added in this way i.e. using the `;uriparams` format. Any header can be added through GHM, including the From header. See chapter 6 Generic Header Manipulation (GHM).

### 3.4 183 to 180 conversion

If you want to convert a *183 Session Progress* (i.e. with SDP) from the ITSP to a *180 Ringing* (i.e. without SDP) towards the PBX add `;cnv183` to the Domain Name or IP Address in the *Service Provider Domain* field at the SIP Trunk page.

**Service Provider Domain:**

```
10.20.30.40;cnv183
```

### 3.5 Do not REGISTER to trunk server(s)

Add `;no-reg` to the Domain Name or IP Address in the *Service Provider Domain* field at the SIP Trunk page. Requires firmware >= 6.0.3.

**Service Provider Domain:**

```
10.20.30.40;no-reg,10.20.30.41
```

This disables sending of REGISTER to a proxy when multiple proxies are entered in the Service Provider Domain.

### 3.6 Do not automatically monitor trunk server(s) with SIP OPTIONS

Add `;no-mon` to the Domain Name or IP Address in the *Service Provider Domain* field at the SIP Trunk page. Requires firmware >= 6.0.3. In firmware >= 6.0.3, SIP Trunk servers are automatically monitored

**Service Provider Domain:**

```
sip.itsp.com,sip2.itsp.com;no-mon
```

This disables sending of SIP OPTIONS to a proxy when multiple proxies are entered in the Service Provider Domain.

### 3.7 B2bua with media via the main dial plan

In order to relay media i.e. to anchor media at the SBC when e.g. releasing or diverting calls back to the operator – this option is synonymous with the “Relay media” trunk page option: on the main dial plan, add `;b2buawm`. A bug in the 5.0.x firmware series prevented this expression from working properly and was fixed in the 6.0.2 firmware. Note, you must have a regexp ($) for the parameter to work. This regexp is used on the main Dial Plan, Forward To rows:

**Forward To …or this… Reg Expr:**

```
$1@10.20.30.40;b2buawm
```
3.8 **Explicitly stating transport**

SIP RFC 3261 specifically states that two URIs are not synonymous if a port or implied parameter is absent in one but present in a second while all other parameters are equal. For implied parameters, such as transport, their presence – or absence – can be important. A UAC must specify its transport when registering to a proxy, if it is not UDP:

Reg Expr:

\[ john@10.20.30.40;transport=tcp \]

Note that RFC3261 deprecates the use of transport=tls although its use on an Ingate is still possible for compatibility. If an ITSP Contact: header contains the transport parameter “transport=tcp” in one response, but removes it in another response – implying “transport=udp” – the transport will be deemed to have changed on the ITSP side, and the Ingate will contact the new destination. If the ITSP cannot handle this transport change (where two independent daemons listen for traffic via the two different transports which may be unaware of each other), this is an error that the ITSP must correct.

3.9 **Specifying Escape Characters (dial string) for e.g. Telia SIP trunk**

Reg Expr:

\[ ;escape-chars=*# \]

3.10 **Explicitly handling only specific METHODS**

Requires firmware >= 4.10.1

Reg Expr – as a tag to a regular expression:

\[ ;methods="ACK,INVITE,BYE,CANCEL,OPTIONS" \]

The above regular expression has the effect that only the named SIP methods ACK, INVITE, BYE, CANCEL and OPTIONS are to be handled.

E.g. (under main dial plan)

Matching RURI:

\[ sip:172.18.137.113@{} \]

Forward To Reg Expr:

\[ 10.3.1.1;methods="OPTIONS" \]

The above expression has the effect that only the SIP method OPTIONS is handled/affected.

E.g. (under SIP Traffic → Routing → User Routing )

User:

\[ alice@proxy.lan \]

Action:

\[ Parallel \]
3.11 Additional information

Here are a few resources we recommend to read more about to build and test regular expressions before they go live:

- [http://www.regular-expressions.info/](http://www.regular-expressions.info/)
- [http://renschler.net/RegexBuilder/](http://renschler.net/RegexBuilder/)
- echo (Test expression) | grep -E (regexp)
4 Conditional Regular Expressions (CRE)

The variable substitution functionality has been enhanced in Ingate, with software version 5.0.4 or later, to allow applying regular expressions and replace on the result of variable substitutions. For example:

`...$(REGMATCH_123.REGMOD_456.REGELSE_(.*).REGMOD_789.REGEND.from.user)...`

first takes the user portion of the From: header, and then performs the regular match/replace which must be enclosed by the tags “REGMATCH_” and “_REGEND” as above:

- the user portion of the From: header is compared with 123. If the user portion contains 123, then the result is modified to 456.
- else, the user portion of the From: header is compared with (.*), which greedily matches everything, so the result of the above expression would be 789.

This was a simplified example just to show the basic logic. In practice this can be much more complex. Note: the examples here are to be embedded in a GHM like: $0?From=$(REGMATCH_123...

4.1 Example CRE - Sweden Number Reduction

The following expression will supply number reduction based on the From: header of a telephone number (user portion) in Sweden:

- If the number starts with (^) a + followed by only digits until its end ($), leave the + and digits as is
- If the number starts with (^) 00 followed by only digits until its end ($), replace 00 with + and leave the digits as they are
- if the number starts with (^) 0 followed by only digits until its end ($), add +46 followed by the digits after the initial 0
- else, supply the number unmodified

The following is just one long line:

`...$.REGMATCH^001([0-9]{10}).REGMOD+[00]$.REGMOD+[0-9]+$.REGMOD+00$1.REGELSE^0$.REGMOD+$1.REGELSE^0$1.REGEND.from.user)...`

Note: The result from the match from each “REGMATCH_” tag respective “REGELSE_” tag is valid only for a replacement in the respectively immediately following “REGMOD_” tag.

4.2 Example CRE - USA Number Reduction

The following expression will supply number reduction based on the From: header of a telephone number in USA:

- If the number already starts with a + followed by only digits, leave the + and digits as they are
- If the number starts with 001 followed by only digits, replace the 001 with +1 and leave the digits as they are
- if the number starts with 1 followed by only digits, add +1 followed by the digits after the initial 1

`...$.REGMATCH^001([0-9]{10}).REGMOD+[001].REGELSE^1([0-9]{10}).REGMOD+1$1.REGELSE^1([0-9]{10}).REGMOD+1$1.REGEND.from.user)...`
4.3 Generic Number Reduction to + prefix for phone numbers 8-12 digits long

Rule

<table>
<thead>
<tr>
<th>Order</th>
<th>Look for the Prefix...</th>
<th>...and change it (add prefix) to</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>00</td>
<td></td>
</tr>
<tr>
<td>2nd</td>
<td>0</td>
<td>(national) e.g. +31</td>
</tr>
<tr>
<td>3rd</td>
<td>+</td>
<td>(no change)</td>
</tr>
<tr>
<td>4th</td>
<td>anything</td>
<td></td>
</tr>
</tbody>
</table>

Regexp

<table>
<thead>
<tr>
<th>Order</th>
<th>Match (and capture)</th>
<th>Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>1st</td>
<td>^00([0-9]{8,12})$</td>
<td>$1</td>
</tr>
<tr>
<td>2nd</td>
<td>^0([0-9]{8,12})$</td>
<td>+31$1</td>
</tr>
<tr>
<td>3rd</td>
<td>^([0-9]{8,12})$</td>
<td>$1</td>
</tr>
<tr>
<td>4th</td>
<td>^.*$</td>
<td>$1</td>
</tr>
</tbody>
</table>

Expression

<table>
<thead>
<tr>
<th>Match</th>
<th>Modify</th>
</tr>
</thead>
<tbody>
<tr>
<td>REGMATCH_^00([0-9]{8,12})$</td>
<td><em>REGMOD</em> $1</td>
</tr>
<tr>
<td>REGELSE_^0([0-9]{8,12})$</td>
<td><em>REGMOD</em> +31$1</td>
</tr>
<tr>
<td>REGELSE_^([0-9]{8,12})$</td>
<td><em>REGMOD</em> $1</td>
</tr>
<tr>
<td>REGELSE_^.*$</td>
<td><em>REGMOD</em> $1</td>
</tr>
<tr>
<td><em>REGEND</em></td>
<td></td>
</tr>
</tbody>
</table>

Expression which operates on user portion of From header

$\{REGMATCH_\^00\([0-9]{8,12}\)$ _REGMOD_ $1$ REGELSE_\^0\([0-9]{8,12}\)$ _REGMOD_ +31$1$ REGELSE_\^\([0-9]{8,12}\)$ _REGMOD_ +\.*$ _REGMOD_ $1$ REGELSE_\^\(+.*\)$ _REGMOD_ +$1$ REGELSE_\^\.*$ _REGMOD_ +$1$ REGENDFROMUSER$\}$

Encoded expression producing a valid From header, operating on user portion of From header

?From=%3csip%3a$\{REGMATCH_\^00\([0-9]{8,12}\)$ _REGMOD_ +$1$ REGELSE_\^0\([0-9]{8,12}\)$ _REGMOD_ +31$1$ REGELSE_\^\([0-9]{8,12}\)$ _REGMOD_ +\.*$ _REGMOD_ +$1$ REGELSE_\^\(+.*\)$ _REGMOD_ +$1$ REGELSE_\^\.*$ _REGMOD_ +$1$ REGENDFROMUSER$\}$%40$(from.host)$%\{([from.uriparams])\}%3e$(from.params)$

Example modifications

<table>
<thead>
<tr>
<th>Phone #</th>
<th>...is modified to...</th>
<th>...due to</th>
<th>reason</th>
</tr>
</thead>
<tbody>
<tr>
<td>911</td>
<td>+911</td>
<td>+.*$ $1</td>
<td>3 digits, no 00, 0 or +</td>
</tr>
<tr>
<td>13009865555</td>
<td>+13009865555</td>
<td>+.*$ $1</td>
<td>11 digits, no 00, 0 or +</td>
</tr>
<tr>
<td>9865555</td>
<td>+9865555</td>
<td>+.*$ $1</td>
<td>7 digits, no 00, 0 or +</td>
</tr>
<tr>
<td>+9865555</td>
<td>+9865555</td>
<td>+.*$ $1</td>
<td>7 digits</td>
</tr>
<tr>
<td>03009865555</td>
<td>+3103009865555</td>
<td>+([0-9]{8,12})$ $1$</td>
<td>10 digits, with 0 prefix</td>
</tr>
<tr>
<td>00800986555</td>
<td>+3100800986555</td>
<td>+([0-9]{8,12})$ $1$</td>
<td>11 digits with 0 prefix</td>
</tr>
</tbody>
</table>

Note: In firmware versions <= 5.0.11 you cannot use $1?From=$(......$1...) – i.e. capture groups on the trunk page (i.e. where $1 is before and after the “?”), and $REGMATCH expressions which also contain $1. The workaround is to use $0?From=$(......$1...). See Errata.
5 Variable Substitution
The Ingate firmware has a range of variables to access all headers and URIs.
Where a SIP URI could be:

```
<sip:user:password@host:port;uriparams>;params
```

Below is a list of the variables used to access URI and header portions for use in substitution:

<table>
<thead>
<tr>
<th>Variable</th>
<th>Explanation</th>
</tr>
</thead>
<tbody>
<tr>
<td>cfg.user</td>
<td>The user part of the Local registrar account matched (if any) by a request</td>
</tr>
<tr>
<td>cfg.host</td>
<td>The host part of the Local registrar account matched (if any) by a request</td>
</tr>
<tr>
<td>ruri.user</td>
<td>User portion of Request-URI</td>
</tr>
<tr>
<td>ruri.host</td>
<td>Host portion of Request-URI</td>
</tr>
<tr>
<td>ruri.uriparams</td>
<td>URI parameters of Request-URI</td>
</tr>
<tr>
<td>header_name.userinfo</td>
<td>The whole user portion, containing password, up to the @ symbol.</td>
</tr>
<tr>
<td>header_name.password</td>
<td>Password of header &quot;header_name&quot; (e.g. to.password)</td>
</tr>
<tr>
<td>header_name.cpassword</td>
<td>As password; but prepended with colon (&quot;;password&quot;) ². Will only expand if header_name.userinfo contains a password.</td>
</tr>
<tr>
<td>header_name.userinfoat</td>
<td>User, Password and Parameters of user portion e.g. “joe:123”, or “joe;tgtp=1...”. In effect, everything between sip(s): and @.</td>
</tr>
<tr>
<td>header_name.host</td>
<td>Host portion of header &quot;header_name&quot; (e.g. contact.host)</td>
</tr>
<tr>
<td>header_name.port</td>
<td>Port portion of header &quot;header_name&quot; (e.g. contact.port)</td>
</tr>
<tr>
<td>header_name.cport</td>
<td>As port, but prepended with a colon “;”. E.g. ?foo=example.com$([from.cport]) Note: Only expands if header_name contains a port param.</td>
</tr>
<tr>
<td>header_name.dname</td>
<td>Display name of header &quot;header_name&quot; (e.g. from.dname)</td>
</tr>
<tr>
<td>header_name.uqdname</td>
<td>Unquoted display name of header &quot;header_name&quot; (e.g. from.uqdname)</td>
</tr>
<tr>
<td>header_name.params</td>
<td>Header parameters of header &quot;header_name&quot; (e.g. contact.params)</td>
</tr>
<tr>
<td>header_name.uriparams</td>
<td>URI parameters of header &quot;header_name&quot; (e.g. contact.uriparams)</td>
</tr>
<tr>
<td>header_name.telnum</td>
<td>Tel-URI telephone number of header &quot;header_name&quot; (e.g. from.telnum)</td>
</tr>
<tr>
<td>header_name.uri</td>
<td>Complete URI of header &quot;header_name&quot; (e.g. contact.Uri)</td>
</tr>
<tr>
<td>hdr.header_name</td>
<td>The complete header value of header &quot;header_name&quot;</td>
</tr>
<tr>
<td>hdr.header_name[2]</td>
<td>The complete header value of 2nd occurrence of header &quot;header_name&quot;</td>
</tr>
<tr>
<td>rawhdr.header_name</td>
<td>The complete header value of header &quot;header_name&quot; (not escaped)</td>
</tr>
<tr>
<td>ip.interface</td>
<td>The IP address of physical network interface &quot;interface&quot;, (e.g. ip.eth1)</td>
</tr>
<tr>
<td>header_name.dnameuri</td>
<td>Display name and complete URI of header &quot;header_name&quot; (e.g. from.dnameuri)</td>
</tr>
</tbody>
</table>

₁ (Available from SW version 5.0.4) ² (Available from SW > 6.0.0)

Where a SIP URI is:

```
<sip:user:password@host:port;uriparams>;params
```

And may be formed by:
$(x.dname) <sip:$(x.user):$(x.password)@$(x.host)$(x.uriparams)糖尿$(x.params)

Or may be formed by:

$((x.dname)) <sip:$$((x.user))$(((x.password)))@((x.host))$$((x.uriparams)))糖尿$$((x.params))

which may be formed by the encoded string:

$((x.dname))%20%3Csip:$($((x.user))$:$(x.password))%40$((x.host))$$((x.uriparams))糖尿$$((x.params))

All variables are invoked by using the $(variable) syntax. The same is achieved by using

$([variable]) – the difference being that, if the content of $([variable]) is empty, then the result is not inserted. If $(variable) syntax is empty, the literal string $(variable) is inserted.

Note: To use any variables, a regular expression match must have been done, together with a regular expression substitution. E.g. $1?header=...

If you just want to rewrite the From header, use the Forward To Reg Expr field and append a ";from=" parameter. The from parameter value may contain references to Reg Expr sub-strings. It may reference both sub-strings in the Matching From Header, and the Matching Request-URI Reg Expr fields. Sub-strings of the From header are referenced as $fx (where x is an integer, 0 or greater). And Request-URI sub-strings are referenced as $rx.

5.1 Port and Password

Parameters password and port are always prepended by a semicolon. If you write an expression...

...$((from.user))%40(from.host)%3a((from.port))... where the from header contained no port parameter, your string evaluates to 123@192.168.1.1: which is an invalid URI, since there is no port value after the colon. If it is unknown in advance whether port or password would be found in a URI of an incoming SIP message, the following CHMs (which require firmware >= 5.0.4) can work around whether or not to insert a semicolon dependent upon the respective parameters’ existence:

$((CONDIF.from.port))$((CONDYES.PLAIN.%3A))$((CONDYES.from.port))

or

$((CONDIF.from.password))$((CONDYES.PLAIN.%3A))$((CONDYES.from.password))

Convenience variables for the above are provided in Ingate firmware >= v6.0:

header_name.cport yields e.g. :5060
header_name.cpassword yields e.g. :p@$word

Regarding indices/indexes [?]
- index [0] is same as no index [], and becomes index 1, and gets the first instance of that header (the one at the top)
- index [1] is the first instance of that header (the one at the top)
- the last index [-1] gets the last instance of that header (at the bottom)
- index [-2] gets the second last instance of that header (the second from the bottom) and so on.

E.g.

Via: SIP/2.0/UDP via[1].host:via[1].port;via[1].uriparams
Via: SIP/2.0/UDP via[-1].host:via[-1].port;via[-1].uriparams

is the same as

Via: SIP/2.0/UDP via.host:via.port;via.uriparams
Via: SIP/2.0/UDP via[-1].host:via[-1].port;via[-1].uriparams

is the same as

Via: SIP/2.0/UDP via[-3].host:via[-3].port;via[-3].uriparams
Via: SIP/2.0/UDP via[-2].host:via[-2].port;via[-2].uriparams
Example INVITE where URIs and headers are written as variables

INVITE sip:ruri.user@ruri.host;ruri.port;ruri.uriparams SIP/2.0
Via: SIP/2.0/UDP via[1].host:via[1].port;via[1].uriparams
Via: SIP/2.0/UDP via[-1].host:via[-1].port;via[-1].uriparams
To: "to.uqdomain" <sip:to.user@to.host>
From: from.dname <sip:from.user@from.host>;from.params
Call-ID: call-id.user@call-id.host
P-Asserted-Identity: sip:P-Asserted-Identity.user@P-Asserted-Identity.host
P-Preferred-Identity: <sip:P-Preferred-Identity.user@P-Preferred-Identity.host>
Contact: <sip:Contact.user@Contact.host>
Record-Route: <sip:Record-Route[1].user@Record-Route[1].host;record-route.uriparams>
Record-Route: <sip:Record-Route[-1].user@Record-Route[-1].host;record-route.uriparams>
Session-Expires: hdr.session-expires
User-Agent: hdr.user-agent
Supported: hdr.supported
Allow: hdr.allow
Max-Forwards: hdr.max-forwards
CSeq: 1 INVITE
Content-Type: application/sdp
Content-Length: ...
...

Example equivalences - From
From: from.dname <sip:from.user@from.host>;from.params
is the same as
From: "from.uqdomain" <sip:from.user@from.host>;from.params
is the same as
From: from.uri
is the same as
From: hdr.from
5.2 Examples

<table>
<thead>
<tr>
<th>Adding From Header</th>
</tr>
</thead>
<tbody>
<tr>
<td>sip:$1@192.168.1.1;from=&quot;sip:$1(from.user)@1.2.3.4&quot;</td>
</tr>
<tr>
<td>sip:$<a href="mailto:1@example.com">1@example.com</a>;from=&quot;sip:$1(from.user)@1.2.3.4&quot;</td>
</tr>
<tr>
<td>from=&quot;sip:$1(from.user)@example.com&quot;</td>
</tr>
<tr>
<td>Replaces the From domain with &quot;example.com&quot;.</td>
</tr>
<tr>
<td>from=&quot;sip:$<a href="mailto:f1@example.com">f1@example.com</a>&quot;</td>
</tr>
<tr>
<td>Replaces the From domain with &quot;example.com&quot; when matching on From Header Reg Expr &quot;(.<em>)@.</em>&quot;.</td>
</tr>
</tbody>
</table>

The examples assume that there is a match done resulting in that $1 contains the user part of the Request-URI.

**Note:** The **From Header** is the only header that can be added or changed as a ;tag i.e. using the ;uriparams format without using a GHM. All headers can be added through GHM, including the From header. See chapter 6 Generic Header Manipulation (GHM).
Generic Header Manipulation (GHM)

The rules for header manipulation of SIP messages are configured in the GUI for Dial Plan and SIP Trunk Page in the same fields as regular expressions are entered. The rules in the "Forward to" fields are applied when there is a match in the corresponding matching fields.

The Generic Header Manipulation uses grammar of the header field components in a SIP URI as described in SIP RFC 3261.

The general form of a SIP URI, is:

```
sip:user:password@host:port;uri-parameters?headers
```

SIP headers are written in a GHM as an encoded URL. That is, they consist of name and value pair of the form `Header_Name=Header_Value`. The first header is written after a `?` and any additional headers are separated by the symbol `&`.

Within the `Header_Value`, characters disallowed according to RFC 3261 must be escaped with a `%HEX` notation where HEX is a 2-digit hexadecimal number representing the escaped character (i.e. URL-Encoded String).

For example the SIP header:

```
Diversion=<sip:7202839130@192.168.1.1>
```

is written as:

```
Diversion=%3csip%3a7202839130%40192.168.1.1%3e
```

The table below lists characters often used, and their corresponding URL encoding:

<table>
<thead>
<tr>
<th>Character</th>
<th>HEX Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>@</td>
<td>%40</td>
</tr>
<tr>
<td>;</td>
<td>%3a</td>
</tr>
<tr>
<td>;</td>
<td>%3b</td>
</tr>
<tr>
<td>,</td>
<td>%2c</td>
</tr>
<tr>
<td>=</td>
<td>%3d</td>
</tr>
<tr>
<td>&lt;</td>
<td>%3c</td>
</tr>
<tr>
<td>&gt;</td>
<td>%3e</td>
</tr>
<tr>
<td>Spaces</td>
<td>+</td>
</tr>
</tbody>
</table>

For GHM, the headers are added at the end of the Regular Expression URI, after "?" which denotes the start of the GHM. Then the Header Name is identified, then a "=" is used to indicate the Header Value, then the Header Value is provided as a URL-Encoded String. "&" is used as a separator for subsequent headers. A header can be removed by assigning the reserved value, "__remove" (two underscores).

"<" and ">" are required in the Header Value under some conditions. Refer to RFC 3261 section 20.10,

Example:

```
R-URI          Header Name        Header Value (URL-Encoded String)
sip:$1@192.168.1.1?Diversion=%3csip%3a7202839130%40192.168.1.1%3e
```

"?" is the Start separator

```
"%HEX" is for escaped characters
```

Adding or Replacing a Header

For GHM, if a specified header field is already present in the incoming SIP request, the header is replaced, if a specified header field is NOT present in the incoming SIP request, the header field is added.
6.1.1 Adding and Replacing Header Examples

**P-Asserted Identity**

- sip:$1@192.168.1.1?P-Asserted-Identity=%3csip%3a7202839130%40192.168.1.1%3e
- sip:$1@example.com?P-Asserted-Identity=%3csip%3a7202839130%40192.168.1.1%3e

**Diversion**

- sip:$1@192.168.1.1?Diversion=%3csip%3a7202839130%40192.168.1.1%3e
- sip:$1@example.com?Diversion=%3csip%3a7202839130%40192.168.1.1%3e

**Privacy**

- sip:$1@192.168.1.1?Diversion=<url-encoded_string>
- sip:$1@example.com?Diversion=<url-encoded_string>

**Multiple**

- sip:$1@192.168.1.1?P-Asserted-Identity=<url-encoded>&Diversion=<url-encoded>
- sip:$1@example.com?P-Asserted-Identity=<url-encoded>&Diversion=<url-encoded>

**Used in combination with regular expression**

- sip:$1@192.168.1.1;b2bua;from="Anonymous@10.182.0.178"?P-Asserted-Identity=<url-encoded_string>&Privacy=id

**Allow header re-write (redundant in >=5.0.6, since SIP methods table handles this)**

- ?Allow=ACK%2cINVITE%2cBYE%2cCANCEL%2cOPTIONS

**Note:** Replace `<url-encoded_string>` with the new header value.

6.2 Removing a header

This is done by starting with the GHM separator “?”, then the Header Name to be removed, followed by “=”, then the reserved value __remove. (two underscores, then "remove")

6.2.1 Examples of Header removal

**Removing Privacy Header**

- sip:$1@192.168.1.1?Privacy=__remove
- sip:$1@example.com?Privacy=__remove
- sip:$(ruri.user)@$ruri.host?Privacy=__remove

6.3 Generic Header Manipulation (GHM) for Responses

GHM for responses works similarly to header manipulation for requests except that header names used for response manipulation need to be prefixed with an exclamation mark “!”. This means that GHM will be performed on response messages, i.e. 180 Ringing, 200 OK, ACK (as opposed to INVITE).

Example to solve Avaya display problem using PAI:
Add P-Asserted Identity Header

| sip:$1@192.168.1.1?!P-Asserted-Identity=$(to.dnameuri) |
| sip:$1@example.com?!P-Asserted-Identity=$(to.dnameuri) |

This adds P-Asserted-Identity to the response with the display name and URI from the To-header.

6.4 Multiple Occurrences of Headers

Header fields are indexed using angle brackets [x] so that one can refer to the n:th occurrence of a certain header.

If a specified header field is present in the incoming SIP request, the header is replaced. Conversely if a specified header field is absent from the incoming SIP request, the header field is NOT added. Replacement operations will only be performed if the header index exists in the message, no headers will be otherwise added. Use [*] as index to refer to all header fields with same name. Use [-1] as index to refer to the last indexed field of a header.

If a variable is unknown or does not exist in an incoming message, the specified header will be left untouched.

6.4.1 Breakout Example

The following is just one long line:

| sip:$0@example.com?User-Agent=$(hdr.user-agent) |
| &Contact=sip%3afoo%40$(ip.eth4)%3buser%3dphone |
| &Organization=mycompany&Privacy=__remove |
| &Diversion[1]=sip%3a$(diversion[1].user)%401.1.1.1 |

Explanation of each Reg Expr component:

sip:$0@example.com
This is the Request URI, $1 is to use the first variable captured from the match, e.g. from Matching Request URI field in the SIP Traffic – Dial Plan page.

?User-Agent=$(hdr.user-agent)
User-Agent header is supplied with the value taken from the incoming User-Agent header, where it exists. The User-Agent header is effectively unchanged (the Ingate itself only rewrites User-Agent header in B2BUA mode).

&Contact=sip%3afoo%40$(ip.eth4)%3buser%3dphone
The Contact Header is replaced with the IP address of eth e.g. Contact: sip:foo@1.2.3.4;user=phone

&Organization=mycompany

&Privacy=__remove
The Privacy header is removed.

&Diversion[1]=sip%3a$(diversion[1].user)%401.1.1.1
Where a Diversion header exists, the host portion is replaced with 1.1.1.1.

6.5 Variable substitution

Variable substitution can be used for GHM as for Regular Expressions. The pre-defined variables can be used in GHM expressions. These variables provide specific values from other headers into the result. See chapter 5 Variable Substitution for a list of the variables that can be used.
### 6.5.1 Variable substitution Examples

<table>
<thead>
<tr>
<th>Modify Contact header with eth3 IP address</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sip:$0@example.com?Contact=sip%3afoo%40$(ip.eth3)%3buser%3dphone</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Add special character + to the From header</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sip:$0@192.168.1.2?from=%3Csip%3a%2B$(from.user)%40172.16.0.1%3E</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduce the From header prefix to USA +1, pass any other URI parameters</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sip:$0?From=%3Csip%3a$(REGMATCH_001([0-9]{10})$ REGMOD_+1$1 REGELSE_1([0-9]{10})$ REGMOD_+1$1 REGELSE ([0-9]{10})$ REGMOD_+1$1 REGELSE (from.user)%40$from.host$ (CONDIF.from.uriparams)$ (CONDYES.from.uriparams) %3E$from.params)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Reduce the From header prefix to USA +1, pass any other URI parameters (alternative)</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sip:$0?From=%3Csip%3a$(REGMATCH_001([0-9]{10})$ REGMOD_+1$1 REGELSE_1([0-9]{10})$ REGMOD_+1$1 REGELSE (from.user)%40$from.host$ ([from.uriparams]) %3E$from.params)</code></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Strip the Display name portion out of the From header</th>
</tr>
</thead>
<tbody>
<tr>
<td><code>sip:$1@192.168.1.1?From=$(from.uri)</code></td>
</tr>
</tbody>
</table>
7 Conditional Variable substitution (CVS)
From Ingate SW version 5.0.4 the variable substitution functionality has been enhanced to allow for conditional variable substitution. It looks like this:

<table>
<thead>
<tr>
<th>Conditional Variable substitution</th>
</tr>
</thead>
<tbody>
<tr>
<td>$(CONDIF.diversion.user)$(CONDYES.diversion.user)$(CONDNO.from.user)</td>
</tr>
</tbody>
</table>

$(CONDIF...)$ evaluates a condition. $(CONDYES...)$ and $(CONDNO...)$ output a result based on the evaluation of that condition.

Example from above: When there is a user portion of a Diversion: header present in the SIP message (e.g. from the PBX), then the above expression supplies the $(diversion.user)$ header portion, otherwise it supplies the $(from.user)$. Note: This CVS evaluates to produce a result, which must be placed in a header. See below.

7.1 Conditional Header Manipulation (CHM)
The purpose of the PLAIN context is to conditionally include URL-encoded plain-text content e.g.:

| ?from=$(CONDIF.diversion.user)$(CONDYES.PLAIN.%3csip%3a12345678%40company.com%3e)$(CONDNO.PLAIN.%3csip%3aABCDEFGH%40company.com%3e) |

When there is a user portion of a Diversion: header, then the From: header produced will be <sip:12345678@company.com>, otherwise it will be <sip:ABCDEFGH@company.com>

Resulting in either:

(CONDYES) ...

Diversion: joe@domain
From: <sip:12345678@company.com>

or

(CONDNO) ...

Diversion: empty or non-existent URI
From: <sip:ABCDEFGH@company.com>

Example. Assuming the above CHM, the following SIP INVITE:

```
INVITE sip:zyx@ingate.com SIP/2.0
Session-Expires: 14400
Via: SIP/2.0/UDP 192.0.2.2:5060;branch=z9hG4bK4cc7285324505f2f5baad12ef7e98054
To: <sip:zyx@ingate.com>
From: <sip:alpha@ingate.com>;tag=2267be7a
Call-ID: 21@sipgt-2d
CSeq: 3 INVITE
Contact: <sip:E4F0pr9vEUD3axDnp@192.0.2.2>
Supported: timer, replaces, path, histinfo
Allow: ACK, CANCEL, INVITE, BYE, NOTIFY, INFO, OPTIONS, PRACK, REFER, UPDATE
Max-Forwards: 15
Content-Type: application/sdp
Content-Length: ...
```

Will evaluate to NO because no Diversion header exists in the INVITE, thus producing:

```
%3csip%3aABCDEFGH%40company.com%3e
```

The expression effectively becomes:

```
?ffrom=%3csip%3aABCDEFGH%40company.com%3e
```

This will result in a modified INVITE once the GHM completes:

```
INVITE sip:zyx@ingate.com SIP/2.0
Session-Expires: 14400
Via: SIP/2.0/UDP 192.0.2.2:5060;branch=z9hG4bK4cc7285324505f2f5baad12ef7e98054
To: <sip:zyx@ingate.com>
From: <sip:ABCDEFGH@company.com>
Call-ID: 21@sipgt-2d
CSeq: 3 INVITE
```
Contact: <sip:E4F0pr9vEUD3axDnp@192.0.2.2>
Supported: timer, replaces, path, histinfo
Allow: ACK, CANCEL, INVITE, BYE, NOTIFY, INFO, OPTIONS, PRACK, REFER, UPDATE
Max-Forwards: 15
Content-Type: application/sdp
Content-Length: ...

The following SIP INVITE:
INVITE sip:zyx@ingate.com SIP/2.0
Session-Expires: 14400
Diversion: hello@123
Via: SIP/2.0/UDP 192.0.2.2:5060;branch=z9hG4bK4cc7285324505f2f5baadi12ef7e98054
To: <sip:zyx@ingate.com>
From: <sip:alpha@ingate.com>;tag=2267be7a
Call-ID: 21@sipgt-2d
CSeq: 3 INVITE
Contact: <sip:E4F0pr9vEUD3axDnp@192.0.2.2>
Supported: timer, replaces, path, histinfo
Allow: ACK, CANCEL, INVITE, BYE, NOTIFY, INFO, OPTIONS, PRACK, REFER, UPDATE
Max-Forwards: 15
Content-Type: application/sdp
Content-Length: ...

Will also evaluate to NO (because no Diversion header exists - a ‘Diversion’ header exists, however), thus producing:
%3csip%3aABCDEFGH%40company.com%3e

The expression again becomes:
?from=%3csip%3aABCDEFGH%40company.com%3e

The following SIP INVITE:
INVITE sip:zyx@ingate.com SIP/2.0
Diversion: gohereinstead@betacorp.xyz;reason=no-answer
Session-Expires: 14400
Via: SIP/2.0/UDP 192.0.2.2:5060;branch=z9hG4bK4cc7285324505f2f5baadi12ef7e98054
To: <sip:zyx@ingate.com>
From: <sip:alpha@ingate.com>;tag=2267be7a
Call-ID: 21@sipgt-2d
CSeq: 3 INVITE
Contact: <sip:E4F0pr9vEUD3axDnp@192.0.2.2>
Supported: timer, replaces, path, histinfo
Allow: ACK, CANCEL, INVITE, BYE, NOTIFY, INFO, OPTIONS, PRACK, REFER, UPDATE
Max-Forwards: 15
Content-Type: application/sdp
Content-Length: ...

Will evaluate to YES (because a Diversion header exists, and contains:
gothereinstead@betacorp.xyz;reason=no-answer), thus producing:
%3csip%3a12345678%40company.com%3e

The expression becomes:
?from=%3csip%3a12345678%40company.com%3e

This will result in a modified INVITE once the GHM completes:
INVITE sip:zyx@ingate.com SIP/2.0
Diversion: gohereinstead@betacorp.xyz;reason=no-answer
Session-Expires: 14400
Via: SIP/2.0/UDP 192.0.2.2:5060;branch=z9hG4bK4cc7285324505f2f5baadi12ef7e98054
To: <sip:zyx@ingate.com>
From: <sip:12345678@company.com>;tag=55566567be7a
Call-ID: 21@sipgt-2d
CSeq: 3 INVITE
Contact: <sip:E4F0pr9vEUD3axDnp@192.0.2.2>
Supported: timer, replaces, path, histinfo
Allow: ACK, CANCEL, INVITE, BYE, NOTIFY, INFO, OPTIONS, PRACK, REFER, UPDATE
Max-Forwards: 15
Content-Type: application/sdp
Content-Length: ...

A note about parameters: we have not included any parameters in our expression. The b2bua will add its own special ;tag parameter separately to the new From header. Any ;tag parameter you copy from the source through the use of a $(from.params) expression will be updated separately by the
b2bua, but other parameters will remain unchanged. If you want to include any parameters found after `<sip:…>` in your result, add `${(from.params)}` to your expression:

```plaintext
?from=$(CONDIF.diversion.user)$(CONDYES.PLAIN.%3csip%3a12345678%40company.com%3e)
$(CONDNO.PLAIN.%3csip%3aABCDEFG%40company.com%3e)${(from.params)}
```

### 7.1.1 More Examples

```plaintext
?from=$(CONDIF.diversion.user)$(CONDYES.PLAIN.%3csip%3a)$(CONDYES.REGMATCH_\^\+([0-9]+)\$REGMOD_+\$1_REGELSE_\^0([0-9]+)\$REGMOD_+\$1_REGELSE_\^0([0-9]+)
\$REGMOD_+46\$1_REGELSE_\.\$REGMOD_+\$1_REGEND.from.user)$(CONDYES.PLAIN.%40company.com%3e)
$(CONDNO.PLAIN.%3csip%3aABCDEFG%40company.com%3e)
```

This does the following:

1. Where the message from the PBX contains a Diversion: header, then the username portion in the From: header (appended with “company.com”) replaces the From: header of the message sent out, after applying the Sweden number Reduction example.
2. Where the message from the PBX does not contain a Diversion: header, no header manipulation is performed.

Resulting in either:

(CONDYES) ...  
Diversion: mike@domain  
From: <sip:+46812345678@company.com>

or an unmodified from header:

(CONDNO) ...  
From: <sip:0812345678@company.com>

### 7.1.2 More Complex Example - History-Info

Where forwarding information is transported via a History-Info: header, then the following should used:

```plaintext
?from=$(CONDIF.history-info[-1].user)$(CONDYES.PLAIN.%3csip%3a)
$(CONDYES.REGMATCH_\^\+([0-9]+)\$REGMOD_+\$1_REGELSE_\^0([0-9]+)\$REGMOD_+
\$1_REGELSE_\^0([0-9]+)\$REGMOD_+46\$1_REGELSE_\.\$REGMOD_+\$1_REGEND.from.user)
$(CONDYES.PLAIN.%40company.com%3e)
```

**Note:** that history-info[-1] refers to the last history-info header present in the SIP message from the PBX.

Resulting in either:

(CONDYES) ...  
History-Info: qwerty  
From: <sip:+46812345678@company.com>

or an unmodified from header:

(CONDNO) ...  
From: <sip:0812345678@company.com>
7.1.3 More Complex Example – Conditional From Header, based on To callee

A company has an old PRI. Outbound calls are now made over the new SIP trunk. On standard calls, they want to send the main number from the PRI out over the SIP trunks. They need to send the real caller ID of the SIP trunks on 911 calls, however.

The old PRI number is 555-101-2001.
Their new SIP trunk number is 555-777-8888.

So, if the call is to 911 i.e.
the user portion of the To header is 911,
the resulting From number is 5557778888,
if the call is not to 911 (i.e. to anything other than 911),
the resulting From number is 5551012001.

A suitable expression would be:

```
?from=$(CONDIF.REGMATCH_^911$_REGEND.to.user)$(CONDYES.PLAIN.%3csip%3a5557778888@192.0.2.2)$(CONDNO.PLAIN.%3csip%3a5551012001@192.0.2.2)
```

Resulting in either (911 is being called):

(CONDYES) ...
From: <sip:5556668888@192.0.2.2>

Or (911 is not being called):

(CONDNO) ...
From: <sip:5551012001@192.0.2.2>

7.2 URI Parameter Chaining

You can also manipulate multiple headers this way, by chaining the header manipulations in the usual way with the “&” character. The example below is like the previous History-Info example above, plus it sets the field P-Preferred-Identity:

```
?from=$(CONDIF.diversion.user)$(CONDYES.PLAIN.%3csip%3a)
$(CONDYES.REGMATCH_^\([0-9]+_REGMOD+$1_REGELSE^0\([0-9]+_REGMOD+$1_REGELSE^3d\)REGELSE^10\([0-9]+_REGMOD+$1_REGELSE_\).*_REGMOD+$1_REGEND)from.user)$(CONDYES.PLAIN.%40company.com)&P-Preferred-Identity=%3csip%3aanother_id%40shop.com%3e
```

Resulting in either:

(CONDYES) ...
Diversion: mike@domain
From: <sip:+46812345678@company.com>
P-Preferred-Identity: <sip:another_id@shop.com>

or an unmodified from header:

(CONDNO) ...
From: <sip:0812345678@company.com>
P-Preferred-Identity: <sip:another_id@shop.com>
**Note:** If you want to add a header (which did not already exist in the message) only under certain circumstances e.g. Replace a possibly existing Referred-By: header with a Diversion: header

```
?Diversion=$(CONDIF.Referred-By.user)$(CONDNO.ABORT)$(CONDYES.PLAIN.%3csip%3a)
 $(CONDYES.Referred-By.user)$(CONDYES.PLAIN.%40)$(CONDYES.ReferredBy.host)
 $(CONDYES.PLAIN.%3e)&Referred-By=__remove
```

If no Referred-By header is present then without the $(CONDNO.ABORT) the following string for header manipulation would remain:

?Diversion=
 &Referred-By=__remove

which would result in an empty diversion header in the resulting message. But with the $(CONDNO.ABORT), the resulting header manipulation string becomes

?Diversion=$(ABORT) &Referred-By=__remove

and since header manipulations with unresolved variables will be skipped, no empty diversion header will be added. In effect becoming just:

?Referred-By=__remove

### 7.2.1 Illegal Chaining

The following example of chaining is illegal, and will not work:

```
?from=$(CONDIF.diversion.user) $(CONDYES.PLAIN.%3csip%3a&P-Preferred-Identity=%3csip%3aanother_id%40shop.com%3e)
```

For the above example to compile, it must be corrected – note the closing parenthesis:

```
?from=$(CONDIF.diversion.user) $(CONDYES.PLAIN.%3csip%3a)&P-Preferred-Identity=%3csip%3aanother_id%40shop.com%3e
```

Summary: Parameter chaining only works outside of the regular expression $()$ context, and cannot be chained conditionally.
8 Keyword / Grammar Summary – CVS, CHM and CRE

8.1 Tests
$([ \text{CONDIF}(.\text{REGMATCH}…)] \mid \text{CONDIFNOT} [ \text{CONDANDIF} [ \text{CONDANDIFNOT} [ \text{CONDORIF} [ \text{CONDORIFNOT} \text{.Header.} \text{Portion}]])])$

8.2 Results
$([\text{CONDYES} [\text{PLAIN.Text} \mid \text{.Header.} \text{Portion}] \mid \text{.ABORT} [\text{CONDNO} [\text{PLAIN.Text} \mid \text{.Header.} \text{Portion}] \mid \text{.ABORT}])$

8.3 RegExp
$([\text{REGMATCH}_{\text{REGMOD}_{\text{REGELSE}_{\text{REGMOD}_{\ldots}}}} \text{.Header.} \text{Portion})$

Note: Keywords and expressions are evaluated from left to right, i.e. forwards in their construction.