

MTAS Network Tracing

MTAS

USER GUIDE

Copyright

© Ericsson AB 2016, 2017. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

Disclaimer

The contents of this document are subject to revision without notice due to continued progress in methodology, design and manufacturing. Ericsson shall have no liability for any error or damage of any kind resulting from the use of this document.

Trademark List

All trademarks mentioned herein are the property of their respective owners. These are shown in the document Trademark Information.



Contents

1	Introduction	1
1.1	Prerequisites	1
2	Tools	3
2.1	NetTrace	3
2.2	AppTrace and AppLog	9
2.3	NetTraceCollector	9
2.4	Machine-readable Min Trace Level	10
2.5	Machine-readable Max Trace Level	11
3	NetTrace Function	13
3.1	NetTrace Procedure	14
3.2	Analysis of Trace Outputs	21





1 Introduction

This document describes the procedure that is to be used to obtain network traces from MTAS.

1.1 Prerequisites

This section describes the prerequisites for this document.

It is assumed that users of this document are familiar with performing operations within the area for O&M in general.

1.1.1 Documents

Before starting this procedure, ensure that the following documents are available:

- *AppTrace User Guide*
- *IMS Common Components Troubleshooting Guide*
- *MTAS Troubleshooting Guideline*

1.1.2 Conditions

Certain troubleshooting activities can have an impact on node performance. For example, trace activation can cause traffic disturbance and is not recommended without first consulting Ericsson. However, NetTrace can be activated for a few users and sessions without adversely affecting performance (up to 10 users is the recommended limit).





2 Tools

This section describes the tools that can be used for network tracing in MTAS.

2.1 NetTrace

Note: The inherent problem with observing the behavior of a system by tracing is the consumed capacity of the tracing itself. If the cost is too high, it can interfere with the primary function of the system, at worst even causing system failure.

NetTrace is a tool that allows the user to trace transactions that traverse the MTAS depending on user-defined filter criteria. These transactions are formatted and output in standardized XML file format (in this document referred to as “machine readable”) according to the 3GPP specification, [3GPP TS 32.423, 8.1.0 :Telecommunication management; Subscriber and equipment trace; Trace data definition and management](#).

Alternatively, traces can be read directly from the AppLog and AppTrace files (referred to as “human-readable”). Human-readable format of traces is proprietary and not specified by [3GPP TS 32.423](#).

It is possible to trace at two levels; Min (minimum) and Max (maximum) for both machine-readable and human-readable output formats.

When active, tracing is performed on all MTAS implemented SIP interfaces.

Descriptions for “ieGroup name” and “ie name” can be found in the [3GPP TS 32.423](#).

The presence of an information element in the following tables is defined by the P (presence) column as follows:

- M = Mandatory. The element is always present.
- O = Optional. The element can be present.

The terms `tracedPublicId1` and `tracedPublicId2` used in the following tables refer to the Public User Ids that triggered the trace. The element `tracedPublicId1` is always present as one Public User Id must have triggered the trace.

If the trace was triggered by more than one Public User Id, it is output as `tracedPublicId2`. One example for this is if a Public User Id were specified as an `OrigPublicId` and a different Public User Id were specified as a `TermPublicId`, and a session was set up between the users. In this case, both Public User Ids are triggering in the same trace.



2.1.1 Machine-readable SIP Output at Min Level

At Min Trace Level, the SIP transactions are represented by several `.xml` tags. Limited information is output when tracing at this level.

For the standard XML elements included in the output, refer to the 3GPP specification [3GPP TS 32.423](#).

The MTAS implemented data output is listed in Table 1 and Table 3.

Table 1 Machine-readable SIP Output Request Data at Min Level

Request			
ieGroup Name	ie Name	Presence	Comment
tracedPublicIds ⁽¹⁾	tracedPublicId1	M	A Public User Id that triggered tracing of this request
tracedPublicIds ⁽¹⁾	tracedPublicId2	O	A Public User Id that, with tracedPublicId1, triggered tracing of this request
-	Request-Line	M	SIP Request line
Message Headers	To	M	SIP To header
Message Headers	From	M	SIP From header
Message Headers	Call-ID	M	SIP Call-ID header
Message Headers	CSeq	M	SIP CSeq header

(1) ieGroup name “tracedPublicIds” is only output once, when the trace session is started.

Table 2 Machine-readable Standard SIP Output Request Data at Min Level

Attribute Name	Description
Standard attributes	Standard Trace Data Definition and Management 3GPP TS 32.423 attributes
fileHeader: fileFormatVersion	This attribute specification identifies the file format version applied by the sender. For example, “32.423 V8.1.0” vendorName=“Ericsson AB”



Attribute Name	Description
traceCollec: beginTime	This attribute specification contains a time stamp that refers to the start of the first trace data that is stored in this file. It is a complete time stamp including day, time, and delta UTC hour. For example, "2010-06-29T08:18:43+01:00".
traceRecSession: traceSessionRef	Attribute specification that provides a unique trace session identifier as described in Trace concepts and requirements 3GPP TS 32.421 . A user-defined identity of the trace session.
traceRecSession: traceRecSession Ref	Attribute specification that provides a unique trace recording session identifier as described in Trace concepts and requirements 3GPP TS 32.421 , and Trace control and configuration management 3GPP TS 32.422 . This attribute is derived from hashing of the Call-ID.
msg function	Attribute specification that provides the function name associated to the traced message. For example, "SIP".
msg name	Attribute specification that provides the function name associated with the traced message. For example, "INVITE".
initiator	Optional element that identifies the Network Element (NE) initiator of the protocol message. For example, [address == 130.100.96.123, port == 50195, transport == Udp].
target	Optional element that identifies the NE target of the protocol message. For example, [address == 10.64.65.10, port == 5082, transport == Udp].
ie	Information elements specific to Ericsson IMS SIP requests.



Attribute Name	Description
tracedPublicId1	A Public User Id that triggered tracing of this request for served user. If tracing is triggered for the originating user only, then it contains the Public Id of originating user. If tracing is triggered for the terminating user only, then it contains the Public Id of the terminating user.
tracedPublicId2	A Public User Id that triggered tracing of this request for terminating user. <code>tracedPublicId2</code> exists together with the <code>tracedPublicId1</code> in case when both originating and terminating users involved in a session are traced. This attribute is optional and exists only when both originating and the terminating user in a session are traced.
Request-Line	SIP Request line
To	SIP To header
From	SIP From header
Call-ID	SIP Call-ID header
CSeq	SIP CSeq header

Table 3 Machine-readable SIP Output Response Data at Min Level

Response			
ieGroup Name	ie Name	Presence	Comment
-	Status-Line	M	SIP Status line
Message Headers	To	M	SIP To header
Message Headers	From	M	SIP From header
Message Headers	Call-ID	M	SIP Call-ID header
Message Headers	CSeq	M	SIP CSeq header



Table 4 Machine-readable Standard SIP Output Response Data at Min Level

Attribute Name	Description
Standard attributes , see Table 2	Standard Trace Data Definition and Management 3GPP TS 32.423 attributes
ie	Information elements specific to Ericsson IMS SIP requests
PublicId1	A Public User Id that triggered tracing of this Response for served user. If tracing is triggered for the originating user only, then it contains the Public Id of originating user. If tracing is triggered for the terminating user only, then it contains the Public Id of the terminating user.
PublicId2	A Public User Id that triggered tracing of this Response for terminating user. <code>tracedPublicId2</code> exists together with the <code>tracedPublicId1</code> in case when both originating and terminating users involved in a session are traced. This attribute is optional and exists only when both originating and the terminating user in a session are traced.
Status-Line	SIP Status line
To	SIP To header
From	SIP From header
Call-ID	SIP Call-ID header
CSeq	SIP CSeq header

2.1.2 Human-readable SIP Output at Min Level

At Min Trace Level, the SIP transactions are represented in plain-text form within the AppTrace. The Message Header and applicable parameters are output on individual lines. Limited information is output when tracing at this level.

The MTAS implemented data output is listed in Table 5 and Table 6.

Table 5 Human-readable SIP Output Request Data at Min Level

Request		
Msg	Presence	Comment
traceSessionRef ⁽¹⁾	M	Indicates the forlop specified used by the operator
origPublicId ⁽¹⁾	M	The Originating Public User Id derived from this request
termPublicId ⁽¹⁾	M	The Terminating Public User Id derived from this request
tracedPublicId1 ⁽¹⁾	M	A Public User Id that triggered tracing of this request
tracedPublicId2 ⁽¹⁾	O	A Public User Id that triggered tracing of this request
Initiator	M	IP/Port/Transport that initiated this request
Target	M	IP/Port/Transport that is the intended recipient of this request
Request-Line	M	SIP Request line
To	M	SIP To header
From	M	SIP From header
Call-ID	M	SIP Call-ID header
CSeq	M	SIP CSeq header

(1) These fields are only output once, when the trace session is started.

Table 6 Human-readable SIP Output Response Data at Min Level

Response		
Msg	Presence	Comment
Initiator	M	IP/Port/Transport that initiated this request
Target	M	IP/Port/Transport that is the intended recipient of this request
Status-Line	M	SIP Status line
To	M	SIP To header



Response		
Msg	Presence	Comment
From	M	SIP From header
Call-ID	M	P Call-ID header
CSeq	M	SIP CSeq header

2.1.3 Machine-Readable SIP Output at Max Level

At Max Trace Level, the SIP transactions are encoded into hexadecimal and output as raw data in .xml file format. In contrast to Min level, the complete contents of each request, command, or response are output.

For the standard XML elements included in the output, refer to [3GPP TS 32.423, 8.1.0: Telecommunication management; Subscriber and equipment trace; Trace data definition and management](#).

Post-processing is required on the generated .xml files to obtain meaningful trace data.

2.1.4 Human-readable SIP Output at Max Level

At Max Trace Level, the SIP transactions are represented in plain-text form within the vDicos AppLog. In contrast to Min level, the complete contents of each request or command/response are output. For more information about vDicos Applog, refer to *vDicos Management*.

2.2 AppTrace and AppLog

vDicos AppTrace is used to realize the NetTrace.

For a detailed description of AppTrace functionality, refer to *AppTrace User Guide*.

For a detailed description of Applog, refer to *MTAS Logs*.

2.3 NetTraceCollector

The `NetTraceCollector` is a Perl-based tool that collects trace data from the AppLog and outputs the data as in .xml format.



2.4 Machine-readable Min Trace Level

At Min trace level, the SIP transactions are represented by several XML elements. Not all SIP headers are represented when tracing at this level.

SIP tracing at Min level is in plaintext and possible to read without post-processing, although post-processing would normally be undertaken.

An example of the XML file output for the Min trace level is provided in Example 1.

Note: The XML file usually contains more than one message.

In Example 1, “==>” is used to highlight where the output lines have been shifted down to fit the PDF version of this document.

```
<?xml version="1.0" encoding="UTF-8"?>
<trace CollecFile xmlns="http://www.3gpp.org/ftp/specs/archive/32_series/==>
32.423#traceData" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ==>
xsi:schemaLocation="http://www.3gpp.org/ftp/specs/archive/32_series/==>
32.423#traceData">
<fileHeader fileFormatVersion="32.423 V8.1.0" vendorName="Ericsson AB">
<fileSender/>
<trace Collec beginTime="2010-06-29T08:18:43+01:00"/>
</fileHeader>
<traceRecSession traceSessionRef="781292" traceRecSessionRef="98765321">
<ue idType="Private User Id" idValue="Not Available"/>
<msg function="SIP" name="INVITE" changeTime="0.000" vendorSpecific="false">
<initiator>{address == 130.100.96.123, port == 50195, transport == ==>
Udp}</initiator>
<target>{address == 10.64.65.10, port == 5082, transport == Udp}</target>
<ie name="Request-Line">INVITE sip:B-TC_SIP2_INV0010@ericsson.com SIP/2.0</ie>
<ieGroup name="Message Headers">
<ie name="CSeq">1 INVITE</ie>
<ie name="To">sip:B-TC_SIP2_INV0010@ericsson.com</ie>
<ie name="Call-ID">12946732</ie>
<ie name="From">sip:A-TC_SIP2_INV0010@ericsson</ie>
</ieGroup>
</msg>
<msg function="SIP" name="100" changeTime="0.002" vendorSpecific="false">
<initiator>{address == 10.64.65.10, port == 5082, transport == Udp}</initiator>
<target>{address == 130.100.96.123, port == 5060, transport == Udp}</target>
<ie name="Request-Line">SIP/2.0 100 Trying</ie>
<ieGroup name="Message Headers">
<ie name="CSeq">1 INVITE</ie>
<ie name="To">sip:B-TC_SIP2_INV0010@ericsson.com;==>
tag=p65544t1277792323m105643c827s1_897840873-1601435024</ie>
<ie name="Call-ID">12946732</ie>
<ie name="From">sip:A-TC_SIP2_INV0010@ericsson</ie>
</ieGroup>
</msg>
```



```
<msg function="SIP" name="INVITE" changeTime="0.007" vendorSpecific="false">
<initiator>{address == 10.64.65.10, port == 5082, transport == Tcp}</initiator>
<target>{address == 130.100.96.123, port == 5061, transport == Tcp}</target>
<ie name="Request-Line">INVITE sip:B-TC_SIP2_INV0010@ericsson.com SIP/2.0</ie>
<ieGroup name="Message Headers">
<ie name="CSeq">1 INVITE</ie>
<ie name="To">sip:B-TC_SIP2_INV0010@ericsson.com</ie>
<ie name="Call-ID">897845640-1095003790</ie>
<ie name="From">sip:A-TC_SIP2_INV0010@ericsson</ie>
</ieGroup>
</msg>
<msg function="SIP" name="180" changeTime="0.018" vendorSpecific="false">
<initiator>{address == 130.100.96.123, port == 61335, transport == ==>
Tcp}</initiator>
<target>{address == 10.64.65.10, port == 5082, transport == Tcp}</target>
<ie name="Request-Line">SIP/2.0 180 Ringing</ie>
<ieGroup name="Message Headers">
<ie name="CSeq">1 INVITE</ie>
<ie name="To">sip:B-TC_SIP2_INV0010@ericsson.com;tag=348714871</ie>
<ie name="Call-ID">897845640-1095003790</ie>
<ie name="From">sip:A-TC_SIP2_INV0010@ericsson</ie>
</ieGroup>

.....

</msg>
</traceRecSession>
</traceCollecFile>
```

Example 1 XML File Output for Min Trace Level

2.5 Machine-readable Max Trace Level

At Max trace level, the SIP transactions are encoded in hexadecimal format and output as raw data. In contrast to the Min level, the complete contents of each request or response are output.

Post-processing of XML files obtained at Max level is required to obtain human readable SIP traces.

An example of the XML file output (partial trace) for the Max trace level is provided in Example 2.

In Example 2, “==>” is used to highlight where the output lines have been shifted down to fit the PDF version of this document.



```
<?xml version="1.0" encoding="UTF-8"?>
<traceCollecFile xmlns="http://www.3gpp.org/ftp/specs/archive/32_series/==>
32.423#traceData" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" ==>
xsi:schemaLocation="http://www.3gpp.org/ftp/specs/archive/32_series/==>
32.423#traceData">
<fileHeader fileFormatVersion="32.423 V8.1.0" vendorName="Ericsson AB">
<fileSender/>
<traceCollec beginTime="2010-07-09T10:56:13+01:00"/>
</fileHeader>
<traceRecSession traceSessionRef="31320" traceRecSessionRef="1234567">
<ue idType="Private User Id" idValue="Not Available"/>
<msg function="SIP" name="INVITE" changeTime="0.000" vendorSpecific="false">
<initiator>{address == 130.100.96.185, port == 24866, transport == Udp}<==>
</initiator>
<target>{address == 130.100.96.185, port == 5082, transport == Udp}</target>
<rawMsg protocol="SIP" version="2.0">494E56495445207369703A42403132372E302E30
2E313A38393031205349502F322E300D0A416C6C6F773A20494E564954452C41434B2C43414E4
3454C2C4259452C5550444154452C4F4B0D0A43616C6C62D49443A206B45627430703954534945
587A786C4F6B7A3668666D5752614F706B6B614951403132372E302E302E310D0A45787069726
5733A2031363030300D0A4D61782D466F7277617264733A2037300D0A435365713A203020494E
564954450D0A546F3A202273697063616C6C62D4222203C7369703A73697063616C6C6C5F6261736
96342403132372E302E302E313A383930313E0D0A46726F6D3A202273697063616C6C6C2D412220
3C7369703A73697063616C6C6C5F626173696341403132372E302E302E313A383830313E3B74616
73D704E7751586C4749574E0D0A502D41737365727465642D4964656E746974793A2022736970
63616C6C62D4122203C7369703A73697063616C6C6C5F626173696341403132372E302E302E313A3
83830313E0D0A502D41737365727465642D4964656E746974793A203C74656C3A2B3132333435
3637383931303E0D0A507269766163793A2069643B73657373696F6E3B757365720D0A436F6E7
46163743A202273697063616C6C6C2D4122203C7369703A73697063616C6C6C5F6261736963414031
32372E302E302E313A383830313E3B713D313B616374696F6E3D70726F78793B6E6F7468696E6
73D353B72656469726563743B657870697265733D350D0A5265636F72642D526F7574653A203C
7369703A3132372E302E302E313A383830313B6C723E0D0A526F7574653A203C7369703A31323
72E302E302E313A353036313B6C723E2C3C7369703A3132372E302E302E313A383930313B6C72
3E0D0A5669613A205349502F322E302F554450203132372E302E302E313A383830313B6272616
E63683D7A39684734624B424D3578664E704C0D0A4D696E2D53453A20333630300D0A53657373
696F6E2D457870697265733A20333630303B7265667265736865723D7561730D0A537570706F7
27465643A2074696D65720D0A436F6E74656E742D446973706F736974696F6E3A207365737369
6F6E0D0A436F6E74656E742D4C656E6774683A203134390D0A436F6E74656E742D547970653A2
06170706C69636174696F6E2F7364700D0A0D0A763D300D0A6F3D412032383930383434353236
203238393038343435323620494E20495034203133342E3133382E35382E3234380D0A733D2D0
D0A633D494E20495034203133342E3133382E35382E3234380D0A743D3020300D0A6D3D766964
656F2032303030205254502F4156502033340D0A623D41533A3132380D0A613D7274706D61703
A333420483236332F383030300D0A0D0A</rawMsg>
</msg>

.....

</traceRecSession>
</traceCollecFile>
```

Example 2 XML File Output (Partial Trace) for Max Trace Level



3 NetTrace Function

The principle of NetTrace is to allow a user the possibility to log SIP transactions traversing the MTAS for fault finding and localization purposes.

Using the vDicos AppTrace, a trace session can be configured to trace SIP transactions based on the filtering of the Originating Public User Ids or Terminating Public User Ids (for both Min and Max levels), or both. The flow of actions required to obtain trace outputs for SIP is shown in Figure 1.

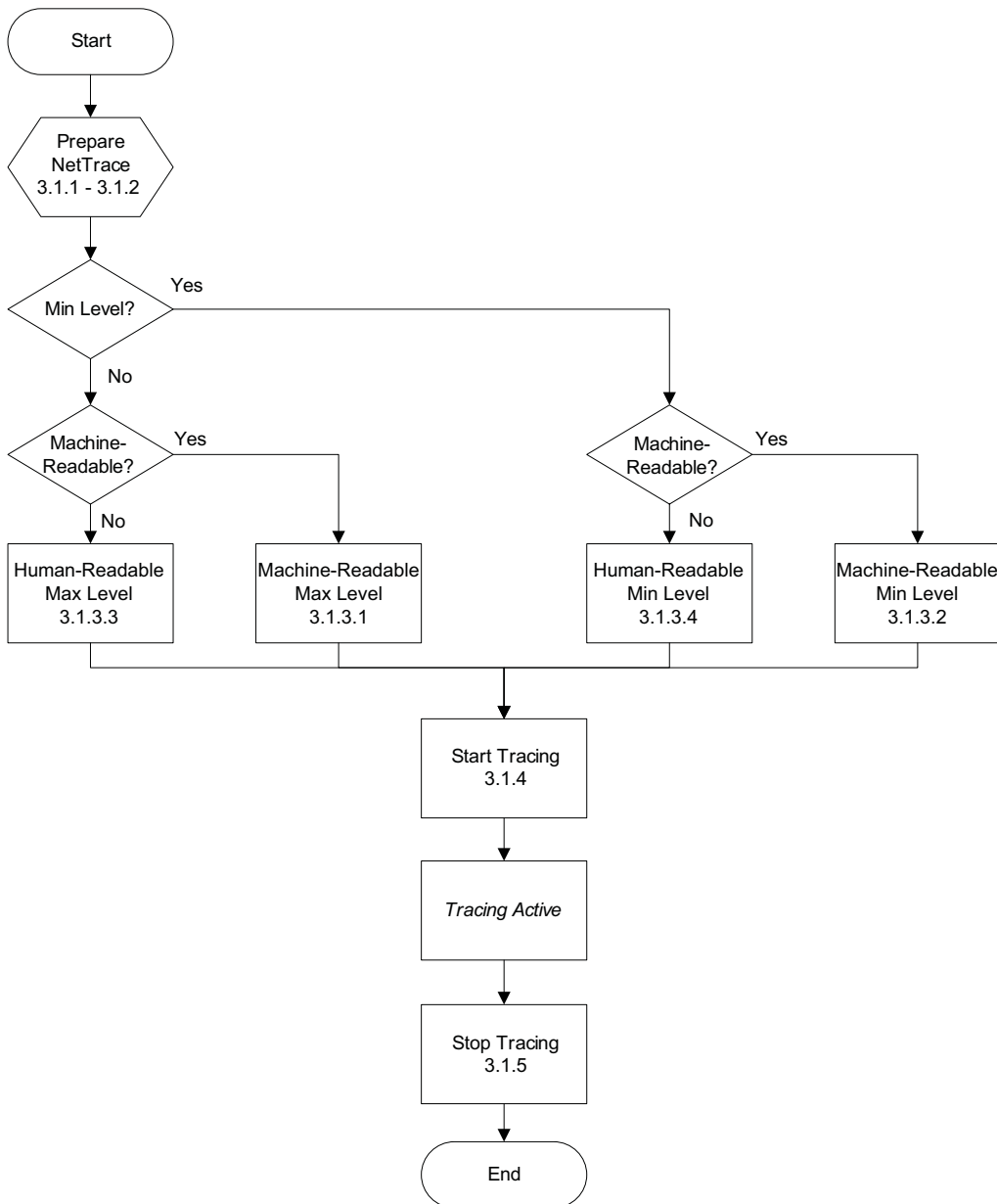


Figure 1 Simplified Flow Configuration and Use of NetTrace

3.1 NetTrace Procedure

This section describes the NetTrace procedure.



3.1.1 Prepare NetTrace

The following sequence is applicable when preparing NetTrace for any type of trace output. “Terminal1” and “Terminal2” refer to two different shells that must be opened.

This procedure is valid when the user has direct access to the node.

“Terminal1” and “Terminal2” refer to two separate System Controllers (shells).

To prepare NetTrace for any type of trace output:

1. For more information on how to ensure that the `.xml` files (if applicable) are output correctly, refer to *IMS Common Components Troubleshooting Guide*.
2. Log on to the primary System Controller SC (from Terminal1), then from the SC log on to one of the Payloads (PLs).

```
> ssh -Y root@<OAM VIP>
```

```
> ssh root@<PL>
```

3. Open a new shell (Terminal2), log on to one of the SCs. From the SC, log on to one of the Payloads (PLs) and start the NetTrace collector (applicable for machine-readable only):

```
> ssh -Y root@<OAM VIP>
```

```
> ssh root@<PL>
```

```
> cd /opt/ericsson/cmco/nettrace/bin
```

Check if the `nettracecollector.pl` script is already running by using, for example:

```
> ps -ef | grep nettracecollector.pl
```

If the `nettracecollector.pl` is not running, start the script:

```
> ./nettracecollector.pl &
```

4. From Terminal1, enter the `AppTrace-CLU` directory:

```
> ssh root@<PL>
```

```
> cd /opt/lpmv/bin/apptrace
```

For more information about AppTrace commands, refer to *AppTrace User Guide*.

5. From Terminal1, gather Trace Domains:

```
> ./collect_domains.sh
```



6. From Terminal1, verify Trace Domains:

```
> ./verify_domains.sh
```

The following MTAS trace domains must be present in the domain tree:

- `ims.mtas.nettrace.init`
- `ims.mtas.nettrace.info`
- `ims.mtas.nettrace.sip`
- `ims.mtas.netio.rx`
- `ims.mtas.netio.tx`
- `ims.mtas.nettrace.rx`
- `ims.mtas.nettrace.tx`

If any of the listed trace domains do not exist, consult the next level of maintenance support.

7. From Terminal1, create a trace session:

```
> ./begin_session.sh
```

8. From Terminal1, include all processors in the trace session:

```
> ./include_processors.sh -a
```

9. From Terminal1, add process types:

```
> ./add_process_type.sh ApplicationProcess.1060633  
> ./add_process_type.sh SipDistributorProcessNew.1126  
603
```

3.1.2 Set up the Traced User

This section describes how to specify the public users to be traced.

The parameter “forlop” is a “Trace Identity” and is a positive integer value from 0 through 1048575 (20 bits). The default value of forlop is zero, which is predefined to mean “anonymous forlop”. The value is chosen by the operator.

To specify the public users to be traced on Terminal 1:

1. Trace the originating user:

```
> ./insert_expression.sh 'ims.mtas.nettrace.init($OrigP  
ublicId == "<PublicIdToBeTraced>") => $forlop = <forlop>'
```



Examples:

- ```
> ./insert_expression.sh 'ims.mtas.nettrace.init(
$OrigPublicId == "sip:userA@domain.x") => $forlop
=12345'
```
- ```
> ./insert_expression.sh 'ims.mtas.nettrace.init($Or
igPublicId == string(sip:userA@domain.x)) => $forlop
=12345'
```

2. Trace the terminating user:

```
> ./insert_expression.sh 'ims.mtas.nettrace.init($TermP
ublicId == "<PublicIdToBeTraced>") => $forlop = <forlop>'
```

Examples:

- ```
> ./insert_expression.sh 'ims.mtas.nettrace.init(
$TermPublicId == "sip:userA@domain.x") => $forlop
=12345'
```
- ```
> ./insert_expression.sh 'ims.mtas.nettrace.init($Te
rmPublicId == string(sip:userB@domain.x)) => $forlop
=12345'
```

3. It is possible to trace multiple users by expressing their Public Ids within the same expression using the OR and AND operators.

Examples:

- ```
> ./insert_expression.sh 'ims.mtas.nettrace.i
nit(($OrigPublicId == "sip:userA@domain.x") &&
($TermPublicId == "sip:userB@domain.y")) =>$forlop
= 12345'
```
- ```
> ./insert_expression.sh 'ims.mtas.nettrace.
init(($OrigPublicId == "sip:userA@domain.x")
|| ($TermPublicId == "sip:userB@domain.y")) =>
$forlop = 12345'
```

Note: For more information on combining logical expressions using OR and AND operators, refer to *AppTrace User Guide*.

Note: Setting up the traced user is a common step that must be performed for all trace level combinations, that is machine/human readable at max/min level.

3.1.3 Set up the Trace Level

This section describes how to set up the applicable trace levels.



3.1.3.1 Machine-readable Max Level

The following sequence is applicable when specifying the trace output is to be machine readable (.xml) at Max level.

To specify the trace output machine readable (.xml) at Max level:

1. The setup of the traced user is performed according to Section 3.1.2 Set up the Traced User on page 16.
2. From Terminal1, insert expressions:

```
> ./insert_expression.sh 'ims.mtas.nettrace.info =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

Note: This is mandatory for both Min and Max levels.

3. From Terminal1, insert expressions for incoming or outgoing SIP traffic, or both:

```
> ./insert_expression.sh 'ims.mtas.nettrace.rx =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

and or

```
> ./insert_expression.sh 'ims.mtas.nettrace.tx =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

Note: For machine-readable Max level trace depth, it is possible to record incoming SIP traffic (using `ims.mtas.nettrace.rx` domain) or outgoing SIP traffic (using `ims.mtas.nettrace.tx` domain) or both (using both `ims.mtas.nettrace.rx` and `ims.mtas.nettrace.tx` domains).

3.1.3.2 Machine-readable Min Level

The following sequence is applicable when specifying the trace output is to be machine readable (.xml) at Min level.

To specify the trace output machine readable (.xml) at Min level:

1. The setup of the traced user is performed according to Section 3.1.2 Set up the Traced User on page 16.
2. From Terminal1, insert expressions:

```
> ./insert_expression.sh 'ims.mtas.nettrace.info =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```



Note: This is mandatory for both Min and Max levels.

3. From Terminal1, insert expressions for incoming and outgoing SIP traffic:

```
> ./insert_expression.sh 'ims.mtas.nettrace.sip =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

3.1.3.3

Human-readable Max Level

The following sequence is applicable when the trace output is to be human-readable at Max level.

To specify the trace output human-readable at Max level:

1. The setup of the traced user is performed according to Section 3.1.2 Set up the Traced User on page 16.
2. From Terminal1, insert expressions:

```
> ./insert_expression.sh 'ims.mtas.netio.info =>L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

Note: This is mandatory for both Min and Max levels.

3. From Terminal1, insert expressions for incoming or outgoing SIP traffic, or both:

```
> ./insert_expression.sh 'ims.mtas.netio.rx =>
L($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

or

```
> ./insert_expression.sh 'ims.mtas.netio.tx =>
L($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

or both

Note: For human readable Max level trace depth, it is possible to record incoming SIP traffic (using `ims.mtas.netio.rx` domain) or outgoing SIP traffic (using `ims.mtas.netio.tx` domain) or both (using both `ims.mtas.netio.rx` and `ims.mtas.netio.tx` domains).

3.1.3.4

Human-readable Min Level

The following sequence is applicable when the trace output is to be human-readable at Min level.

To specify the trace output human-readable at Min level:



1. The setup of the traced user is performed according to Section 3.1.2 Set up the Traced User on page 16.

2. From Terminal1, insert expressions:

```
> ./insert_expression.sh 'ims.mtas.netio.info => L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

Note: This is mandatory for both Min and Max levels.

3. From Terminal1, insert expressions for incoming and outgoing SIP traffic:

```
> ./insert_expression.sh 'ims.mtas.netio.sip => L
($processorname, $pid, $date, $time, $id, $forlop,
$MiId, $MiVer, $Length, $Msg)'
```

3.1.4 Start NetTrace

To start NetTrace:

1. From Terminal1, direct output to the AppLog:

```
> ./route_output.sh applog
```

2. From Terminal1, upload the trace session:

```
> ./upload_session.sh
```

3. From Terminal1, start the trace session:

```
> ./start_trace.sh 24
```

The trace session is now active and tracing starts when the trace criteria are fulfilled.

3.1.5 Stop NetTrace

To stop NetTrace:

1. From Terminal1, stop the trace session:

```
> ./stop_trace.sh
```

2. From Terminal1, unload (uninstall) the trace session:

```
> ./unload_session.sh
```

3. From Terminal1, end the trace session:

```
> ./end_session.sh
```




3.2 Analysis of Trace Outputs

This section describes the analysis of trace outputs.

3.2.1 Machine-readable Traces

NetTrace .xml files are accessed from the PL, and are located in:

```
/cluster/storage/no-backup/cmco_utils-cxp9020686/nettrace/mtas
```

The naming convention is as follows:

```
A<date>.<time>-MTAS.<TraceSessionRef>.<TraceRecSessionRef>.xml
```

Where:

- `<date>` is in the form `yyyymmdd`.
- `<time>` is in the form `hhmm`.
- `<TraceSessionRef>` (TSR) is the user-defined forlop Id. User can be OSS-RC or any troubleshooter.
- `<TraceRecSessionRef>` (TRSR) is the system-defined Trace Recording Session Ref.

MTAS calculates the value of hashed Call-ID and assigns it to TRSR.

For example:

```
A20110128.1609-MTAS.jambala.1111.56032.xml
```

Post-processing is required by a system compliant with [3GPP TS 32.423](#).

3.2.2 Human-readable Traces

Human readable traces are accessed from the PL, and are located as AppTrace files in the following directory:

```
/cluster/storage/no-backup/coremw/var/log/saflog/MTASApp  
Logs/vdicos/
```