

Diameter Troubleshooting Guideline

vDicos

TROUBLESHOOTING

Copyright

© Ericsson AB 2015–2018. 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.



Contents

1	Overview	1
2	Diameter Stack	3
2.1	Trace	3
	Reference List	13





1 Overview

This document describes how to troubleshoot the vDicos and CBA Diameter Base component. For troubleshooting, the Diameter stack trace is used.



2 Diameter Stack

This section provides information on Diameter stack.

2.1 Trace

The Diameter stack trace is based on the vDicos Application trace. The Application trace mechanism is a service available in the vDicos runtime environment.

With Application trace, the Diameter stack can be selectively and safely traced. This means that you can choose precisely what to trace on. The trace session itself does not cause any serious disturbance to the system. However, if you set the trace on too many processors and processes, this can cause noticeable degradation in the performance of the system.

To be able to choose what to trace on, the trace domains and the trace levels defined for the Diameter stack must be known. The processors and the processes that the Diameter stack code is running on or in must also be known.

2.1.1 Trace Domains

The top domain for the Diameter stack is `tsp.diameter`. The `tsp.diameter` domain is abstract, that is, it does not define any parameters.

The trace domains under the top domain `tsp.diameter` are listed in Table 1.

Table 1 Trace Domains for `tsp.diameter`

Trace on Parameter Method	Trace on Parameters Method and Message
tsp.diameter.provider	tsp.diameter.provider.core
	tsp.diameter.provider.router
	tsp.diameter.provider.connection_mgr
	tsp.diameter.provider.transport_listener
	tsp.diameter.provider.transport_layer
	tsp.diameter.provider.dlg_transport
	tsp.diameter.provider.traffic



Trace on Parameter Method	Trace on Parameters Method and Message
tsp.diameter.user	tsp.diameter.user.core
	tsp.diameter.user.enc_dec
	tsp.diameter.user.traffic
tsp.diameter.data	tsp.diameter.data.installer
	tsp.diameter.data.model
	tsp.diameter.data.types
tsp.diameter.fw_utils	tsp.diameter.fw_utils.utils
	tsp.diameter.fw_utils.dlg_user
	tsp.diameter.fw_utils.dlg_provider
	tsp.diameter.fw_utils.sharedmemory
tsp.diameter.oam	tsp.diameter.oam.pm
	tsp.diameter.oam.fm
	tsp.diameter.oam.events

Tracing on the Method parameter only shows trace when methods are started and ended.

Tracing on the Method and Message parameters shows trace when methods are started and ended. Also, many trace messages inside the methods are shown. The amount of messages depends on the chosen trace level.

The different domains trace on the following:

— `tsp.diameter.provider`

Mainly runs in the Handler process, but some parts also run in the Transport Listener process (`tsp.diameter.provider.transport_listener` and `tsp.diameter.provider.transport_layer`) and in the Connection Manager process (`tsp.diameter.provider.connection_mgr`).

Outgoing and incoming connections and also all messages related to the base protocol are handled. The `tsp.diameter.provider.traffic` is used for ProviderCore to trace incoming and outgoing messages as arrays of octets, that is, data as it is received from or sent to the transport layer. Messages that are processed by ProviderCore itself, that is, capabilities exchange, peer disconnection, and device watchdog messages, are traced in the `tsp.diameter.provider.traffic`.

— `tsp.diameter.user`

Mainly runs in the Application Service User process. It handles all messages not related to the base protocol. Decoded messages are logged at the `tsp.diameter.user.traffic`.

— `tsp.diameter.data`

`tsp.diameter.data.installer` runs in the Application Installer process. The `tsp.diameter.data.model` runs in all other processes when accessing the database during traffic and in the JIM Server process when accessing the database during provisioning. This part handles the installation of data and all database transactions.

— `tsp.diameter.fw_utils`

Mainly runs in the Handler process and in the Application Service User process. The dialogue handling between these processes is also shown.

— `tsp.diameter.oam`

Runs in all processes. It handles all functionality related to fault and performance management.

— `tsp.diameter.fw_utils`

Mainly runs in the Handler process and in the Application Service User process. The dialogue handling between these processes is also shown. This domain is also used for trace in processes which use Diameter shared memory.

2.1.2 Trace Levels

The trace levels used by Diameter stack are shown in Table 2.

Note: The maximum trace level that can be specified is trace level 64, refer to AppTrace User Guide, Reference [1]. However, only the four trace groups present in Table 2 are implemented in the vDicos Diameter application.

If a trace group has to be printed in the log, increase the trace level of that group by 1. Meaning, that if trace level 55 needs to be printed in the log, specify 56 during the trace session activation.

Table 2 Diameter Trace Levels

Diameter Trace Level	Diameter Trace Level Name	TSP Predefined Equivalent	Trace Level
60	TRACE_ALL	-	All traces, including function start and end traces



Diameter Trace Level	Diameter Trace Level Name	TSP Predefined Equivalent	Trace Level
55	TRACE	DEEAT_DEBUG	Detailed debug traces, including frequent traces
39	INFO_ALL	DEEAT_MINR	Not frequent debug traces
37	INFO	-	Important information
23	WARNING	DEEAT_MAJR	Recoverable errors
7	CRITICAL	DEEAT_CRIT	Critical errors

2.1.3 Processes

The Diameter stack code not only runs in processes owned by the stack itself but also in processes owned by the Application.

The processes owned by the Diameter stack and by the Diameter Application are shown in Table 3.

Table 3 Diameter Stack and Diameter Application Processes

Process Owner	Process	Comment
Diameter stack	DIA_PRC_ConnectionMgrProcess	A process that initiates transport connections to neighbor peers.
	DIA_PRC_TransportListenerProcess	A process used for transport connections initiated from other neighbor peers.
	DIA_PRC_HandlerProcess	A process that handles a specific connection.
	DIA_PRC_DBAccessHandlerProcess	A process used for writing persistent objects to Diameter shared memory.
Diameter Application	Application Installer process	For the correct names of the Application process, refer to application-specific documentation.
	Application Service User process	

Another process that is used when the Diameter code runs is the JIM_ServerProcess. This process is used to access a database.



2.1.4 Processors

All processes owned by the Diameter stack are allocated to the `DIASharedProcPool` pool.

On how the Diameter Application processes are allocated, refer to the application-specific documentation.

2.1.5 Tracing

To create a useful trace, scope the trace. The scope consists of the following:

- Set of processors
- Set of processes
- Set of trace domains
- Trace level

A trace point is turned on only if the following is included in the trace session:

- The processor on which it is executing
- The process in which it is executing
- The trace domain to which it belongs
- The trace level to which it belongs

When the trace session is defined, you can reduce the effect of the trace on the system by setting up a trace program (a filter). Filtering is performed by trace expressions. When executing a trace session, the parameters included in the trace domain are the output together with the `domain_identity` system parameter. The `domain_identity` is the only system parameter output by default, all other system parameters can be selected for output. This is described in AppTrace User Guide, Reference [1].

For more information on how to define a trace session and a trace program, refer to AppTrace User Guide, Reference [1].

2.1.5.1 Example of a Trace Session

The following example shows a trace session task list that traces the `tsp.diameter.provider.core` and `tsp.diameter.user.core` domains, in the processes `DIA_PRC_ConnectionMgrProcess`, `DIA_PRC_TransportListenerProcess`, `DIA_PRC_HandlerProcess`, `TST_ARC_CliInstallerProc`, and the `TST_CSA_ClientServiceProc` process on `Processor1` and `Processor2`.

`TST_ARC_CliInstallerProc` and `TST_CSA_ClientServiceProc` are the application-specific processes for the test application that includes code from the



Diameter stack. All trace points defined in the `tsp.diameter.user.core` and `tsp.diameter.provider.core` trace domains are logged. In this example, the Method and the Message parameters together with the default system parameters are the output.

1. Prepare for a trace session.
 - a Connect to one of the control nodes of the target system:

```
> ssh root@<ip_address>
```
 - b Connect to one of the payload nodes of the target system (depending on pool configuration, appttrace scripts can be not accessible from SCs):

```
> ssh PL-3
```
 - c Change to the directory holding the AppTrace utilities:

```
> cd /opt/lpmsv/bin/appttrace
```
2. Collect and verify domains:

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

```
> ./ls_domains.sh
```

```
> ./verify_domains.sh
```
3. Define the trace session.
 - a Begin:

```
> ./begin_session.sh
```
 - b Add processors:

```
> ./include_processors.sh Processor1
```

```
> ./include_processors.sh Processor2
```
 - c Add process types:

```
> ./add_process_type.sh DIA_PRC_ConnectionMgrProcess.1015860
```

```
> ./add_process_type.sh DIA_PRC_TransportListenerProcess.1015881
```

```
> ./add_process_type.sh DIA_PRC_HandlerProcess.1015878
```

```
> ./add_process_type.sh TST_CSA_ClientServiceProc.542141
```

```
> ./add_process_type.sh TST_ARC_CliInstallerProc.542030
```
 - d Add trace domains:



- ```
> ./insert_expression.sh tsp.diameter.user.core
```
- ```
> ./insert_expression.sh tsp.diameter.provider.core
```
- e Upload the trace session:
- ```
> ./upload_session.sh
```
4. Execute the trace session:
- ```
> ./start_trace.sh 56
```
5. Stop and clean up the trace session:
- ```
> ./stop_trace.sh
```
- ```
> ./unload_session.sh
```

If the trace session is tracing the `tsp.diameter.provider` domain, all trace points defined for this domain and its underlying subdomains are traced. In this example, only the `Method` parameter is logged together with the default system parameters because this is the only parameter that is defined for the `tsp.diameter.provider` domain.

2.1.6 SS7 Tracing and Logging

Diameter uses the SS7 implementation of the SCTP protocol, and employs SS7 Common Parts (SS7CP) communication facilities to interact with SCTP front ends (FEs). SS7 has its own tracing and logging facility provided by SS7CP—therefore, when troubleshooting SCTP related problems, it is often needed to investigate the `ss7trace.log` file. This file contains messages between Diameter and SCTP FEs, along with events logged by the Diameter SCTP transport layer. SS7CP has the following 10 logging levels:

- EINSS7_EMERG 0 /* system is unusable */
- EINSS7_ALERT 1 /* action must be taken immediately */
- EINSS7_CRIT 2 /* critical conditions */
- EINSS7_ERR 3 /* error conditions */
- EINSS7_WARNING 4 /* warning conditions */
- EINSS7_NOTICE 5 /* normal but significant condition */
- EINSS7_INFO 6 /* informational */
- EINSS7_DEBUG_HI 7 /* debug-level messages */
- EINSS7_DEBUG 8 /* debug-level messages */
- EINSS7_DEBUG_LO 9 /* debug-level messages */



There are two types of processes in Diameter that use SS7CP: DIA_PRC_TransportListenerProcess and DIA_PRC_HandlerProcess. The listener process acts as an SCTP server and listens for incoming SCTP associations for all configured Diameter stack instances. The handler process acts as an SCTP client and serves a particular Diameter connection. Each instance of either of these two process types has its own SS7CP user. Each Diameter connection corresponds to a particular SS7CP user. Each Diameter SCTP listener corresponds to a particular SS7CP user; however, one listener listens for incoming connections for all Diameter stack instances. This means that we can enable and disable SS7 tracing and set SS7 logging level separately for each Diameter connection—whereas configuring these things for the Diameter listener can only be configured globally, that is, for all stack instances at once. Page 10 contains attributes for configuring SS7 tracing and logging for Diameter.

Table 4 Attributes for Configuring SS7 Tracing and Logging for Diameter

MO class	Attribute	Values	Description
DIA-CFG-Con figuration	traceSctpLi stener	TRUE, FALSE	Defines whether SS7 traces for all SCTP connection listeners are enabled or disabled.
DIA-CFG-Con figuration	traceSctpHa ndler	TRUE, FALSE	Defines whether SS7 traces for all SCTP connection handlers are enabled or disabled.
DIA-CFG-Con figuration	sctpListene rLogLevel	0–9	Log level for all SCTP connection listeners.
DIA-CFG-Con figuration	sctpHandler LogLevel	0–9	Log level for all SCTP connection handlers.
DIA-CFG-Own NodeConfig	traceSctpHa ndler	TRUE, FALSE, DEFAULT	Defines whether SS7 traces for all SCTP connection handlers that serve connections with the own node is enabled or disabled.
DIA-CFG-Own NodeConfig	sctpHandler LogLevel	0–9, DEFA ULT	Log level for all SCTP connection handlers that serve connections with the own node.
DIA-CFG-Nei ghbourNode	traceSctpHa ndler	TRUE, FALSE, DEFAULT	Defines whether SS7 traces for all SCTP connection handlers that serve connections with the neighbor node.
DIA-CFG-Nei ghbourNode	sctpHandler LogLevel	0–9, DEFA ULT	Log level for all SCTP connection handlers that serve connections with the neighbor node.



MO class	Attribute	Values	Description
DIA-CFG-Con n	traceSctpHa ndler	TRUE, FALSE, DEFAULT	Defines whether SS7 traces for an SCTP connection handler that serves the connection are enabled or disabled.
DIA-CFG-Con n	sctpHandler LogLevel	0–9, DEFA ULT	Log level for an SCTP connection handler that serves the connection.

SS7 tracing and logging parameters for Diameter handlers are configured in the cascading way: the special attribute value, DEFAULT, is used to let the parent MOs define the configuration. Page 11 depicts how enabling and disabling tracing configuration for a handler serving a particular connection is determined. The logging level is configured similarly. Such a cascading configuration way is not applicable to Diameter listeners, as each listener has one common SS7CP user for all stack instances.

Table 5 Configuring Tracing and Logging Parameters

Value of traceSctpHandler attribute in different MOs				
DIA-CFG-Con n	DIA-CFG-Neigh bourNode	DIA-CFG-Own NodeConfig	DIA-CFG -Configur ation	Resulting value
TRUE	Does not matter	Does not matter	Does not matter	TRUE
FALSE				FALSE
DEFAULT	TRUE			TRUE
	FALSE			FALSE
	DEFAULT	TRUE		TRUE
		FALSE		FALSE
		DEFAULT	TRUE	TRUE
			FALSE	FALSE





Reference List

Documents

- [1] AppTrace User Guide, 1/1553-CXP 904 0373