

MTAS Subscriber Data Management Guide

MTAS

USER GUIDE

Copyright

© Ericsson AB 2016–2018. 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.

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	Overview	3
2.1	MTAS Subscriber Data Function	3
2.2	MTAS Subscriber Data Management Function	3
2.3	Diameter Stack and Sh Interface	5
2.4	Delay Sh Queries	15
3	Sh Diameter Stack Configuration	17
3.1	Diameter Stack Configuration Attributes	21
3.2	Diameter Stack Configuration Examples	26
4	Configure Sh Interface	43
5	Caching Contact Data and UE Terminal Type Classification Configuration	47
6	Rebalancing	51
6.1	Activation	51
6.2	Deactivation	51
7	Wholesale for Subscriber Data Configuration	53
8	Examples, Subscriber Data Management	55
8.1	Subscriber Data	55
8.2	Query or Purge Examples	56





1 Introduction

This document describes how to configure the Subscriber Data function in the MTAS.

1.1 Prerequisites

This section describes the prerequisites that must be fulfilled. It is assumed that the user of this document is familiar with the Operation and Maintenance (O&M) area, in general.

1.1.1 Documents

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

- *Ericsson Command-Line Interface User Guide*
- *Diameter Management*
- *Managed Object Model (MOM)*

1.1.2 Conditions

The following condition must apply:

- An Ericsson Command-Line Interface (ECLI) session in Exec mode is in progress.





2 Overview

This section describes the MTAS Subscriber Data function, the MTAS Subscriber Data Management Function, the Diameter stack, and the Sh interface.

2.1 MTAS Subscriber Data Function

The MTAS Subscriber Data function tracks the following:

- The service data of each subscriber.
- The service data of each MMTel service profile.

The function retrieves subscriber service data, the service data of MMTel service profile, and group service data from the Home Subscriber Server (HSS) node, and caches the data for performance reasons. Caching contact data is enabled by default. Disabling caching can be performed by setting CM attribute `mtasSubsDataCacheContactData` to 0.

Service data XML instances have either normalized entries, or non-normalized entries with ad-hoc presentation and CDIV digits present.

For information on normalized entries, refer to *Managed Object Model (MOM)*.

If the administrative state for SCC AS is unlocked and the HSS-based ATCF info storage is enabled (`mtasSubsDataSccAtcfInfoInHss` attribute set to 1), the Subscriber Data function stores the ATCF registration information of the registered user in HSS as transparent data. The ATCF registration information can then be retrieved from the HSS in failover scenarios.

2.2 MTAS Subscriber Data Management Function

The MTAS Subscriber Data management function provides the operator the ability to query and purge subscriber data cached in the MTAS and to place the result of the query or purge into a file which can be retrieved by the operator. The file location is defined by the read-only `mtasSubsDataMgmtFile` attribute.

An option exists for operator to purge the data related to the ongoing call sessions of the subscriber in MTAS by extending the command syntax described in Section 2.2.2 MTAS, Subscriber Data Management Command Syntax on page 4 with an optional string “`#rmSessionData`” at the end. In this case, the session data that the MTAS services have cached for handling ongoing call sessions of related subscribers is deleted besides the subscribers' cached service data. This option is introduced as a workaround in situations where MTAS internal session handling appears to be stuck because of



unhandled errors. The operator needs to use this option with care as this can cause external session-related resources in other IMS nodes (such as Charging Server or media resource handler) to be temporarily stuck. The operator must also know that the MTAS does not immediately remove these external resources after an extended purge operation or even after the session has ended.

For the MTAS Subscriber Data Managed Objects (MOs) *MtasSubsData* and *MtasSubsDataMgmt*, refer to *Managed Object Model (MOM)*.

2.2.1 MTAS Subscriber Data Management for Unregistered Users

The MTAS subscriber data management function manages the data for unregistered subscribers. MTAS handles all identities in an Implicit Registration Set (IRS) as belonging to the same profile. The MTAS only caches one set of the transparent data (subscriber service data) per IRS. The subscriber data is purged on deregistration timer (*mtasSubsDataDeregTimer*) expiry after a call is terminated for an unregistered user. The default value is 6 hours. Keep the value of CM attribute *mtasSubsDataDeregTimer* such that new subscriber data including IRS changes are downloaded from HSS at least once a day (during night).

2.2.2 MTAS, Subscriber Data Management Command Syntax

Query and purge functionality can be used after an HSS zone reload (purge all), before or after upgrade (purge all), maintenance (partial query and purge), and dimensioning (partial purge, migrating existing users to a new MTAS). If a purge for all subscribers (or most subscribers) is needed, it is recommended to start the purge operation when the CPU load on the Traffic Processors is less than 30% (non traffic-intensive period). Compared to when started during a traffic-intensive period (>30%) the operation finishes faster and less of the ongoing traffic is affected. Also, ensure that no query or purge operation is ongoing during the MTAS upgrade procedure, as such operations are not guaranteed to finish successfully and as thus are not allowed (ensure that the *mtasSubsDataMgmtStatus* attribute value is *FINISHED(0)* before starting an upgrade).

The query and purge operation can be executed or stopped by the following actions of *MtasSubsDataMgmt* MO:

- *mtasSubsDataMgmtRunPurge* (*mtasSubsDataMgmtRunPurgePui*);
- *mtasSubsDataMgmtRunQuery* (*mtasSubsDataMgmtRunQueryPui*);
- *mtasSubsDataMgmtStop*();

The *mtasSubsDataMgmtRunQueryPui* and *mtasSubsDataMgmtRunPurgePui* parameters define the filter for the scope of the operation. Syntax is shown in Table 1.



Table 1 Query or Purge Action Parameter Syntax

mtasSubsDataMgmtRunQueryPui or mtasSubsDataMgmtRunPurgePui	Description
sip:someuser@someplace.com or tel:+46123456789	Query or Purge using single URI using sip or tel. The SIP URI is input in canonical format without URI parameters.
sip:*xxxxxx or tel:*xxxxxx	Query or Purge using wildcard
sip:someuser@someplace.com#rmSessionData or tel:+46123456789#rmSessionData	Query or Purge using single URI and command extension using sip or tel. The SIP URI is input in canonical format without URI parameters. It is a workaround in situations where MTAS internal session handling appears to be stuck.
*	Query or Purge all
sip:* or tel:*	Query or Purge "sip:*" or "tel:*" using wildcard (1)
sip:*xxx* or tel:*xxx*	Query or Purge using two wildcards "*"
sip:*xxx*xxx* or tel:*xxx*xxx*	Query or Purge using multiple wildcards "*"

(1) Two wildcards "*" cannot be used next to one another.

2.2.3 Query or Purge File Retrieval

The files produced for the query or purge operations for administration analysis are retrieved using the file transfer configuration to clean up the files created during the query or purge operations. For more information about file transfer, refer to *File Management*.

2.3 Diameter Stack and Sh Interface

The configuration of the Sh/Dh interface on an MTAS node involves defining Diameter stack attributes for Diameter message routing and defining the realm (basically the domain) to which the nodes that handle Sh/Dh messages belong – HSS, Subscriber Location Function (SLF), and Diameter Routing Agent (DRA). Optionally, the configuration involves specifying the hostname of the HSS node to be used by MTAS.

The *MtasSubsData* MO controls the Subscriber Data function for a complete MTAS node.



The configuration of the Diameter stack and the Sh interface of the Subscriber Data function is shared with the XML Document Management Server (XDMS) function.

The configuration of the Number Normalization data of the XDMS function is shared with the Subscriber Data function. For more information, refer to *Managed Object Model (MOM)*.

2.3.1 Supported HSS Network Architectures

MTAS supports the following four deployable HSS network architectures:

- **Classic Single**

Classic HSS network architecture with a single monolithic HSS instance, see Section 2.3.1.1 Classic HSS Network Architecture with a Single Monolithic HSS Instance on page 6.

- **Classic SLF REDIRECT**

Classic HSS network architecture with multiple SLFs in REDIRECT mode and multiple HSS instances, see Section 2.3.1.2 Classic HSS Network Architecture with Multiple SLFs in REDIRECT Mode and Multiple HSS Instances on page 7.

- **Classic SLF PROXY**

Classic HSS network architecture with multiple SLFs in PROXY mode and multiple HSS instances, see Section 2.3.1.3 Classic HSS Network Architecture with Multiple SLFs in PROXY Mode and Multiple HSS Instances on page 10.

- **Layered**

HSS-FE deployment or HSS data layered network architecture, see Section 2.3.1.4 HSS-FE Deployment or HSS Data Layered Network Architecture on page 12.

2.3.1.1 Classic HSS Network Architecture with a Single Monolithic HSS Instance

MTAS communicates directly with a single HSS instance, which stores user-data along with some user-state information (for example, user level subscriptions to service data change notifications). All the Sh requests originating from MTAS go directly to this HSS instance, see Figure 1.



Figure 1 Layout of Classic Single Network Architecture

The Destination-Realm and Destination-Host AVPs are both present in all Sh requests sent by MTAS. The value for both these AVPs are configured in MTAS with configuration attributes `mtasShIfDestinationRealm` and `mtasIfDestinationHost` respectively (see Section 4 on page 43), as shown in Figure 2.

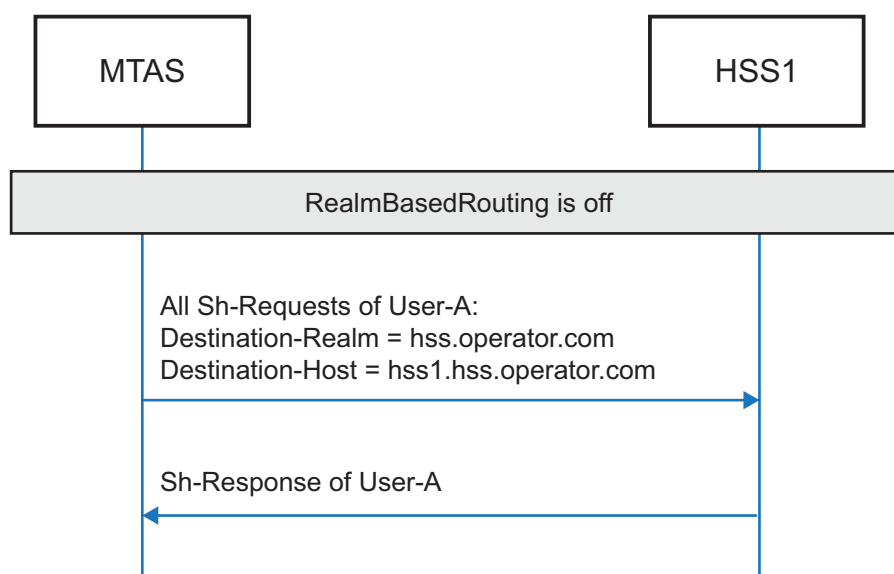


Figure 2 Sh Message Flow for Classic Single Network Architecture

For an example configuration of the Diameter stack of the MTAS node corresponding this architecture, see Section 3.2.1 Example Diameter Stack Configuration for Classic HSS Network Architecture with a Single Monolithic HSS Instance on page 27.

2.3.1.2

Classic HSS Network Architecture with Multiple SLFs in REDIRECT Mode and Multiple HSS Instances

MTAS communicates with multiple SLF instances (in REDIRECT mode) and multiple HSS instances. The HSS instances in this network architecture store

user-data along with some user-state information (for example, user level subscriptions to service data change notifications), see Figure 3.

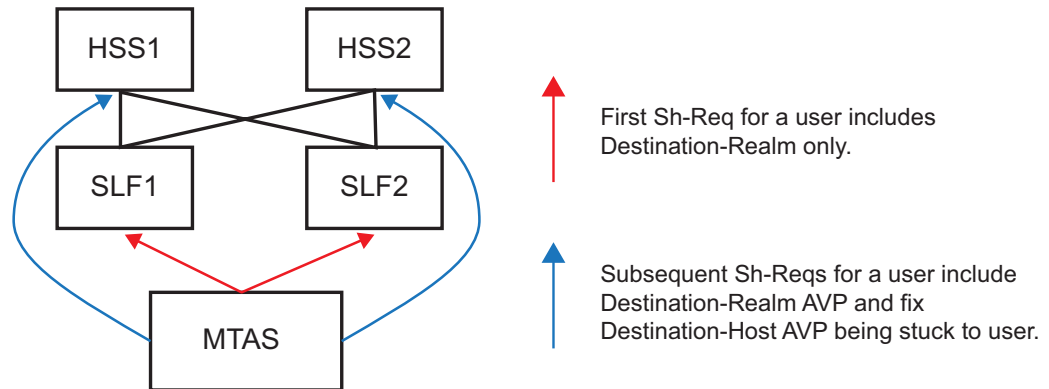


Figure 3 Layout of Classic SLF REDIRECT Network Architecture

MTAS sends the first Sh request of a user to one of the SLFs with no Destination-Host AVP. The SLF rejects the Sh request with a redirect indication answer, containing the hostname of the HSS instance that is serving that user in the Redirect-Host AVP. MTAS sends out the original request again directly towards the HSS node indicated in the redirect answer with the Destination-Host AVP set to the hostname of the indicated HSS instance. If the indicated node is not an HSS but another SLF that answers with another redirect answer again (multiple redirection, not shown in Figure 3 and Figure 4), MTAS sends out the original request again to the new indicated node. MTAS eventually learns the correct HSS hostname to send all subsequent Sh requests to from the Originating-Host AVP of the first successful Sh response that comes from the HSS, as shown in Figure 4.

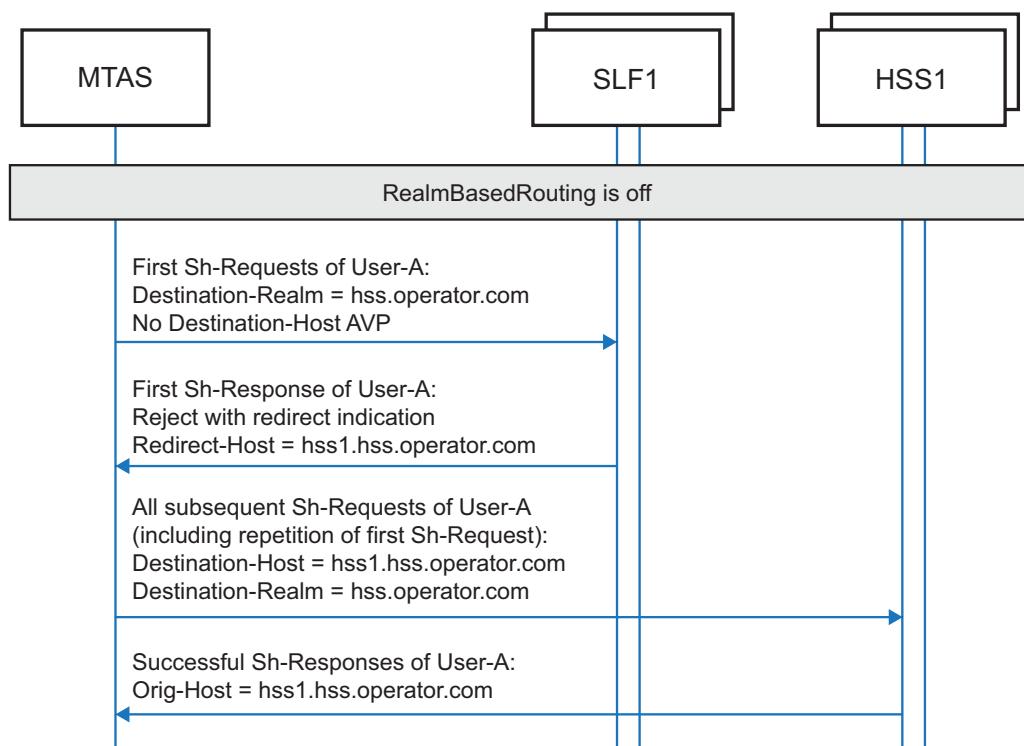


Figure 4 Sh Message Flow for Classic SLF REDIRECT Network Architecture

The Destination-Realm AVP is present in all Sh requests originating from MTAS. The value for this AVP is configured in MTAS with configuration attribute `mtasShIfDestinationRealm` (see Section 4 on page 43).

The realm value in the Destination-Realm AVP determines the Diameter realm which the Diameter stack of the MTAS node uses to randomly select a target SLF from for the first Sh request.

The Destination-Host AVP is only present from the second Sh request onwards and its value is automatically set by MTAS to the hostname of the peer node indicated in the redirect answers and finally in the first successful answer for the initial Sh request. The setting of the `mtasShIfDestinationHost` MTAS configuration attribute is not necessary in this case, and is left empty (see Section 4 on page 43).

The direct routing of all the requests from the second Sh request onwards is based on the value of the Destination-Host AVP.

For configuration of the Diameter stack of the MTAS node corresponding to this architecture, see Section 3.2.2 Example Diameter Stack Configuration for Classic HSS Network Architecture with Multiple SLFs in REDIRECT Mode and Multiple HSS Instances on page 28.

2.3.1.3

Classic HSS Network Architecture with Multiple SLFs in PROXY Mode and Multiple HSS Instances

MTAS communicates with multiple SLF instances (in PROXY mode) and multiple HSS instances. The HSS instances in this network architecture store user-data along with some user-state information (for example, user level subscriptions to service data change notifications), see Figure 5.

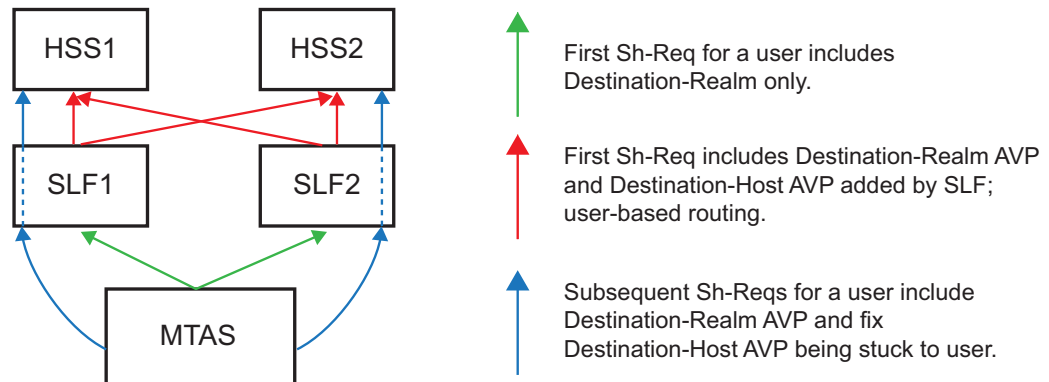


Figure 5 Layout of Classic SLF PROXY Network Architecture

MTAS sends the first Sh request of a user to one of the SLFs with no Destination-Host AVP. The SLF forwards the Sh request to one of the HSS instances that is serving that user with a Destination-Host AVP added by the SLF. The HSS instance answers with a regular successful Sh answer that contains an Originating-Host AVP with the hostname of the HSS instance. The value of this Originating-Host AVP is learned and used by MTAS in all subsequent Sh requests as the value of the Destination-Host AVP. MTAS sends subsequent Sh requests of the user to a randomly selected SLF and the selected SLF forwards the requests to the HSS indicated in the Destination-Host AVP of the requests. All Sh requests/answers between MTAS and the HSSs go through the SLFs in this case, as shown in Figure 6.

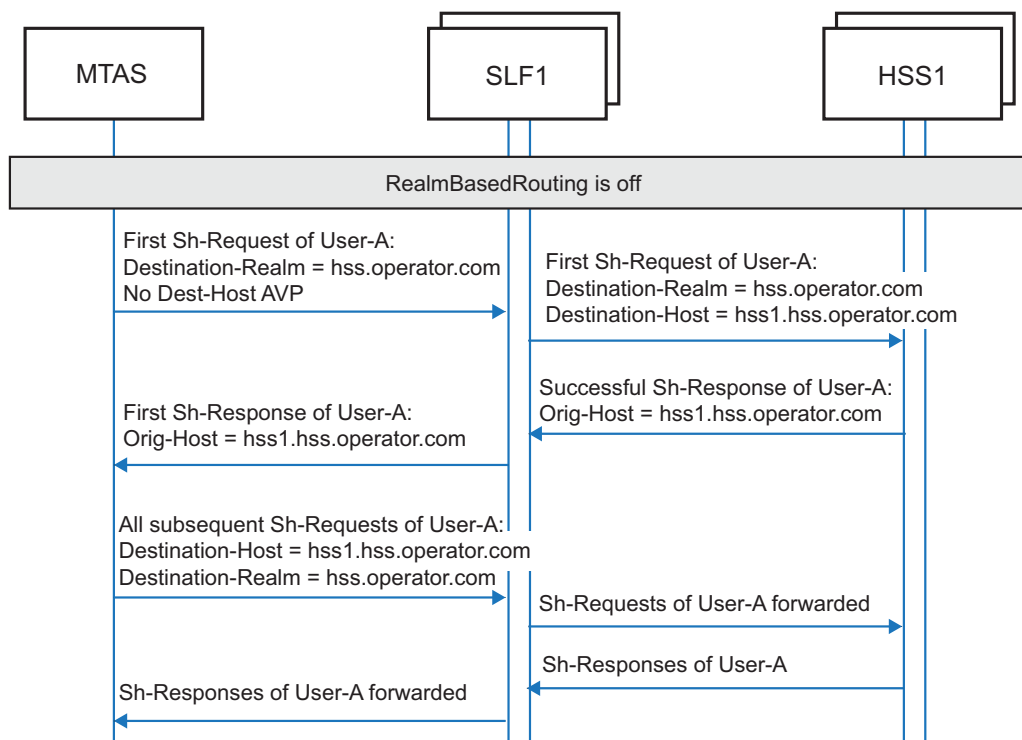


Figure 6 Sh Message Flow for Classic SLF PROXY Network Architecture

The Destination-Realm AVP is present in all Sh requests originating from MTAS. The value of this AVP is configured in MTAS with configuration attribute `mtasShIfDestinationRealm` (see Section 4 on page 43).

The realm value in the Destination-Realm AVP determines the Diameter realm which the Diameter stack of the MTAS node uses to randomly select a target SLF from for the first Sh request.

The Destination-Host AVP is only present from the second Sh request onwards and its value is set by MTAS to the hostname of the HSS instance indicated in the Originating-Host AVP in the first successful Sh answer for the user. The setting of the `mtasShIfDestinationHost` MTAS configuration attribute is not necessary in this case, and is left empty (see Section 4 on page 43).

The routing of all the requests from the second Sh request onwards is based on the value of the Destination-Host AVP.

The Diameter stack in the MTAS node is configured so that the Diameter routing table that contains the peer nodes (*DIA-CFG-NeighbourNode*) routes all the HSS instance destination hostnames possibly present in the Destination-Host AVPs to the IP addresses of the SLFs.

For the configuration of the Diameter stack of the MTAS node corresponding to this architecture, see Section 3.2.3 Example Diameter Stack Configuration for Classic HSS Network Architecture with Multiple SLFs in PROXY Mode and Multiple HSS Instances on page 32.

2.3.1.4

HSS-FE Deployment or HSS Data Layered Network Architecture

Instead of HSS entities, each storing user-data and user-state information, the HSS functionality is deployed separately into a User Data Repository (UDR) (also know as; HSS back-end or external database) storing user-data and user-state information and several HSS Front-Ends (HSS-FEs), which do not store user-related information. HSS-FEs only have load balancing functionality for the Sh requests. The DRAs implement sophisticated LB algorithms, see Figure 7.

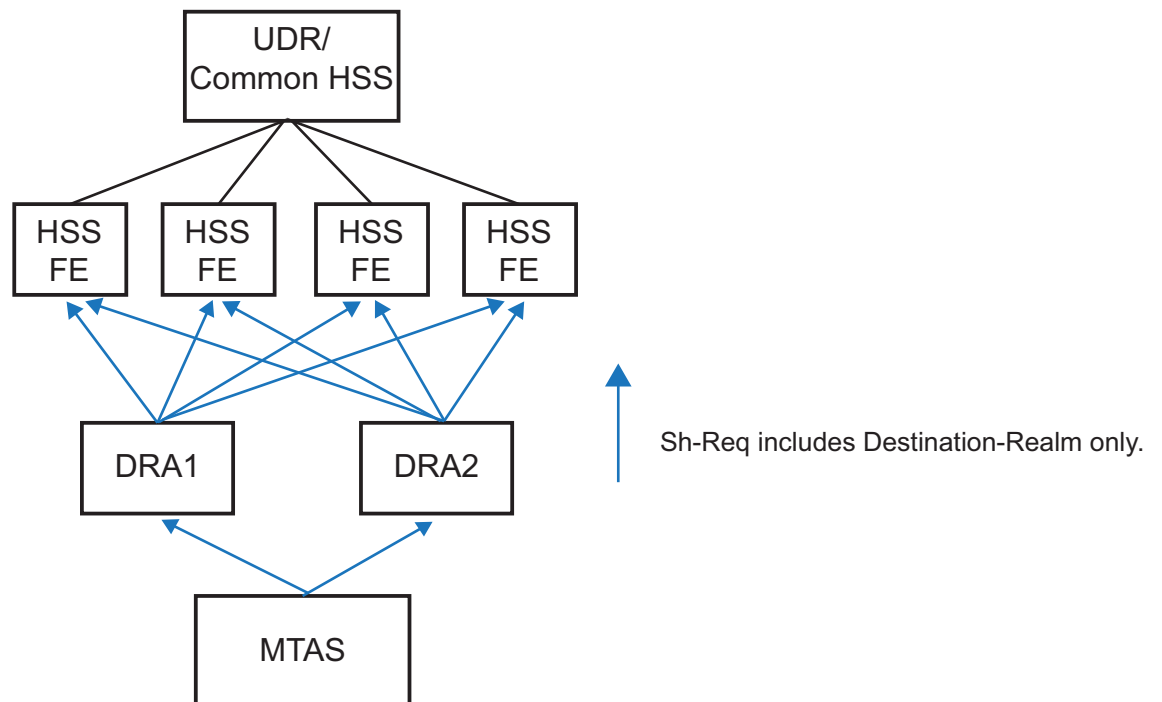


Figure 7 Layout of Layered Network Architecture

MTAS sends all Sh requests towards the neighboring DRAs without Destination-Host AVPs. Only the Destination-Realm AVP is filled in the Sh requests according to the `mtasShIfDestinationRealm` MTAS configuration attribute (see Section 4 on page 43), as shown in Figure 8.

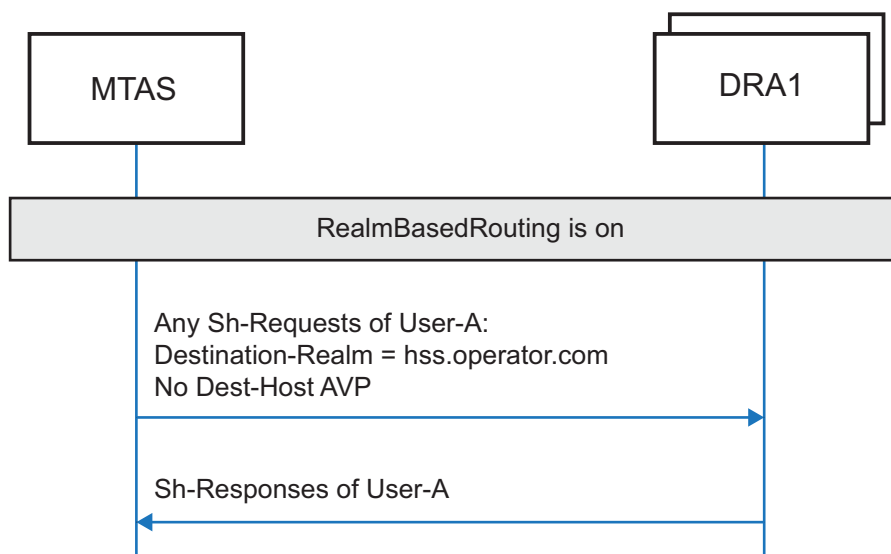


Figure 8 Sh Message Flow for Layered Network Architecture

The Diameter stack of the MTAS node selects a target DRA for an Sh request randomly from the Diameter realm indicated by the Destination-Realm AVP. The DRA then routes the request to one of the HSS-FEs considering the actual load of the HSS-FEs. The HSS-FE finally routes the request to the UDR.

In this case, the benefit for using DRAs is that the DRAs can distribute the Diameter load more efficiently between the HSS-FEs based on additional load and capacity information from HSS-FEs.

The Diameter stack in the MTAS node must be configured so that the Diameter Realm Routing Table configuration (see Section 3.1.3 Add Realm Routing Table Data on page 25) contains the hostnames of all the neighboring DRAs.

For the configuration of the Diameter stack of the MTAS node corresponding to this architecture, see Section 3.2.4 Example Diameter Stack Configuration for HSS-FE Deployment or HSS Data Layered Network Architecture on page 38.

2.3.2 Routing Mechanisms in MTAS

Depending on configuration, the messages on the Sh interface can be routed differently. MTAS supports the following two routing mechanisms:

- Destination host-based request routing (default), used for the following network architectures:
 - Classic HSS network architecture with a single monolithic HSS instance, see Section 2.3.1.1 Classic HSS Network Architecture with a Single Monolithic HSS Instance on page 6.
 - Classic HSS network architecture with multiple SLFs in REDIRECT mode and multiple HSS instances, see Section 2.3.1.2 Classic HSS

Network Architecture with Multiple SLFs in REDIRECT Mode and Multiple HSS Instances on page 7.

- Classic HSS network architecture with multiple SLFs in PROXY mode and multiple HSS instances, see Section 2.3.1.3 Classic HSS Network Architecture with Multiple SLFs in PROXY Mode and Multiple HSS Instances on page 10.
- Realm Routing Table based request routing, used for HSS-FE deployment or HSS data layered network architecture, see Section 2.3.1.4 HSS-FE Deployment or HSS Data Layered Network Architecture on page 12.

2.3.3 Destination Host Based Request Routing

This routing mechanism is used for Classic Single, Classic SLF REDIRECT, and Classic SLF PROXY network architectures. Routing of the messages is done by using the domain and host, that is, the Destination-Realm and Destination-Host AVPs are both considered. The MTAS application configuration attribute `mtasShIfRealmBasedRouting` (see Section 4 on page 43) must be set to 0 for the network architectures using this routing mechanism.

For the network architectures Classic Single, Classic SLF REDIRECT, and Classic SLF PROXY, the HSS nodes store user-state information so it is essential that all Sh-Requests of a user goes to the same HSS instance. In other words, the user needs to be stuck to an HSS instance (also called “user stickiness”).

The destination address based request routing mechanism is used to help achieve the user stickiness. In Sh traffic messages, this means the following:

- In the first Sh request for a given user, the Destination-Host AVP is added to the Sh message by MTAS for the Classic Single network architecture, but not added for the Classic SLF REDIRECT and Classic SLF PROXY network architecture. In either case, the Sh request finds its way to an HSS instance.
- In all subsequent Sh requests, the Destination-Host AVP is added either with a fix hostname value (Classic Single network architecture) or with a value learned from the first successful Sh answer (network architectures Classic SLF REDIRECT and Classic SLF PROXY). The Destination-Host AVP sent by MTAS assures that all the subsequent Sh requests go to the same HSS instance as the first successfully answered request.

This Diameter routing mechanism mostly uses the Diameter routing table that contains the peer node information (DIA-CFG-NeighbourNode) for routing based on the Destination-Host AVP. The Diameter Realm Routing Table is only configured in case of Classic SLF REDIRECT and Classic SLF PROXY network architectures with the addresses of the SLF nodes to be able to route the first Sh requests of the users to the SLFs based on the Destination-Realm AVP.



For more information on Diameter stack routing configurations, see Section 3 on page 17.

2.3.4 Realm Routing Table Based Request Routing

This MTAS routing mechanism is used for the HSS-FE deployment / HSS data layered network architecture. The MTAS application configuration attribute `mtasShIfRealmBasedRouting` (see Section 4 on page 43) must be set to 1 for the network architecture using this routing mechanism.

Routing of Diameter messages is done by using only the realm domain in contrast to the hostname of peer nodes, that is, only the Destination-Realm AVP is considered for all Sh requests, the Destination-Host AVP is not even present in the Sh requests. In this case, the Diameter Realm Routing Table is configured with all the neighboring DRA nodes in the given realm. The Diameter stack selects a DRA node from the Realm Routing Table randomly and it routes the request to the selected node.

For more information on Diameter stack routing configurations, see Section 3 on page 17.

In this case, it is not essential to route all the Sh requests of a given user to the same HSS-FE instance as the HSS-FEs do not store any user-state information. The Sh requests are eventually routed to the same common UDR instance that handles user-data and state information, but this routing is not influenced by MTAS at all.

2.4 Delay Sh Queries

The MTAS Subscriber Data function normally fetches and caches user data from HSS when a user initially registers in the IMS network. This can cause high network load when high amounts of registration are performed at the same time, for example, after a network disturbance event. MTAS supports a mechanism to delay this fetching and caching of data until the first call attempt of the subscriber. As call attempts are more evenly distributed in time, this can help distribute HSS load in time. MTAS can be configured both to delay Sh fetching in every case or only delay when HSS is in overloaded state.

To enable Sh query delay, set `mtasSubsDataInitRegHSSFetchDelay` CM attribute to one of the following values:

- 1 (Delay all queries except ATCF-related queries): This setting is used when Service Centralization and Continuity Application Server (SCC AS) is used with Single Radio Voice Call Continuity (SRVCC) Release 10 to be compatible with Access Transfer Control Function (ATCF) anchoring procedures. Only the data fetching related to ATCF anchoring are performed during initial registration, other data fetching from HSS are delayed to the subscriber's first attempt to make or receiving a call.



- 2 (All HSS data fetching delayed, including ATCF-related ones as well): This setting is used to delay all the Sh requests until the first call attempts of the subscriber. This behavior can cause incompatible behavior with ATCF anchoring when used with SRVCC Release 10 procedures. It cannot be used together with the HSS-based ATCF info storage (`mtasSubsDataSccAtcfInfoInHss` attribute set to 1).
- 3 (HSS data fetching delayed except ATCF related but fallback to delaying all in HSS overload situation is enabled): This setting is used when request only delayed in HSS overload condition. If this setting is used, then MTAS HSS overload timer must be configured with `mtasSubsDataHSSOverloadTimer`. It cannot be used together with the HSS-based ATCF info storage (`mtasSubsDataSccAtcfInfoInHss` attribute set to 1).

For more details on attributes, refer to *Managed Object Model (MOM)*.



3 Sh Diameter Stack Configuration

The configuration of the Sh Diameter stack for the Sh and Dh interfaces specifies attributes for the HSS, SLF, and DRA. In Figure 9, these nodes are defined within the `hss.operator.com` realm. Multiple SLF nodes can coexist with multiple HSS and DRA nodes.

Some MTAS application-specific configuration attributes must be configured in correspondence with the Diameter stack configuration attributes:

- By convention, if only one HSS resides in the network, then the SLF is not used by MTAS/XDMS and the MTAS application configuration attribute `mtasShIfDestinationHost` must be set to the HSS hostname, for example, `hss1.hss.operator.com`.
- If the `mtasShIfDestinationHost` configuration attribute is set, then the network assumption is that only one HSS exists and there is no need for an SLF query. The `mtasShIfDestinationHost` is always set to the hostname of the HSS, for example, `hss1.hss.operator.com`.
- If the `mtasShIfDestinationHost` configuration attribute is left blank, then the network assumption is that more than one HSS exists and there is a need for an SLF query to find the correct HSS.
- If the MTAS is configured for Realm Routing Table based request routing, then the `mtasShIfDestinationHost` configuration attribute is ignored.

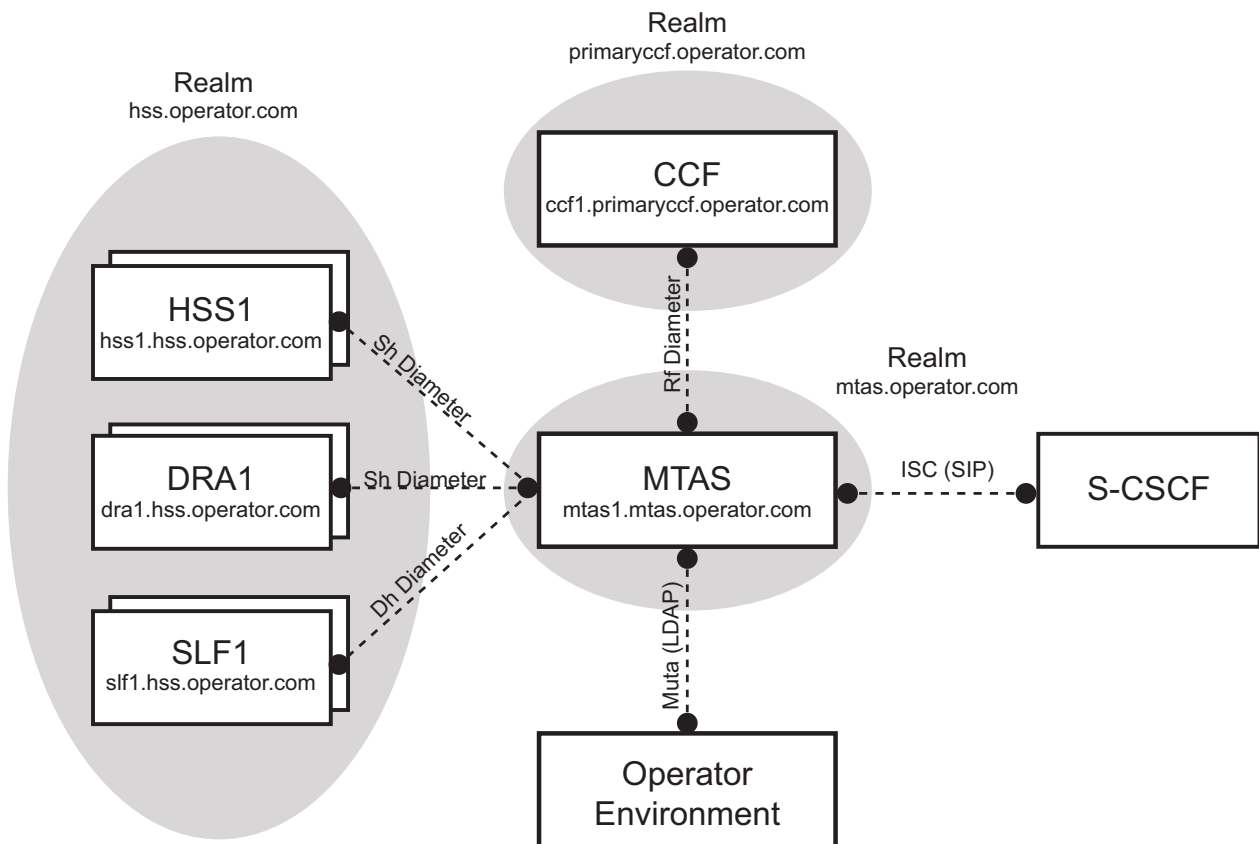


Figure 9 Realm Overview

The following instruction is a general instruction, describing the minimal configuration when configuring the MTAS node, as shown in Figure 9. For configuration examples for the different HSS network architectures, see Section 3.2 Diameter Stack Configuration Examples on page 26.

Note: For more information on how to configure the MOs described in this procedure, refer to *Managed Object Model (MOM)*.

To configure the Diameter stack for the Sh/Dh interfaces:

1. Navigate to the *DIA-CFG-StackContainer* MO (representing a Diameter stack instance).
2. Create two Diameter stack instances (*DIA-CFG-StackContainer*) with stack-names (attribute `stackContainerId`) `MTAS_SH` and `MTASXDMS`.

Stack instance with stack-name `MTAS_SH` is used for the Subscriber Data function of MTAS, used during traffic scenarios.

Stack instance with stack-name `MTASXDMS` is used for the XDMS provisioning function of MTAS, used for provisioning scenarios.



The following steps operate on MO instances below the two stack MO instances created in Step 2 and must be repeated for both stack instances (with some minimal differences indicated).

3. Set the own node configuration attributes. Configure the *DIA-CFG-OwnNodeConfig* MO as described in Table 2.
4. Set the neighbor node configuration attributes. Configure the *DIA-CFG-NeighbourNode* MO, as described in Table 3. This procedure is repeated for each node in the realm.

The Diameter stack must be configured with information about other nodes in the network and information about how to behave when routing messages.

This information is found in the following MOs. These MOs are all part of the Realm Routing Table (RRT) configuration of the Diameter stack and which are in parent-child hierarchy in the configuration attribute tree (from top to bottom):

- *DIA-CFG-Drt*
 - *DIA-CFG-AuthReqContainer*
 - *DIA-CFG-AppRouting*
5. Configure the *DIA-CFG-Drt* MO. The *DIA-CFG-Drt* configuration is described in Table 5.
 6. Set the Authorization Application Routing Configuration attributes. Configure the *DIA-CFG-AppRouting* MO. Perform the Application Routing Configuration, as described in Table 6.
 7. Perform a backup. For more information, refer to *Create Backup*.

See Figure 10 for a browser view example of Diameter stack attributes.

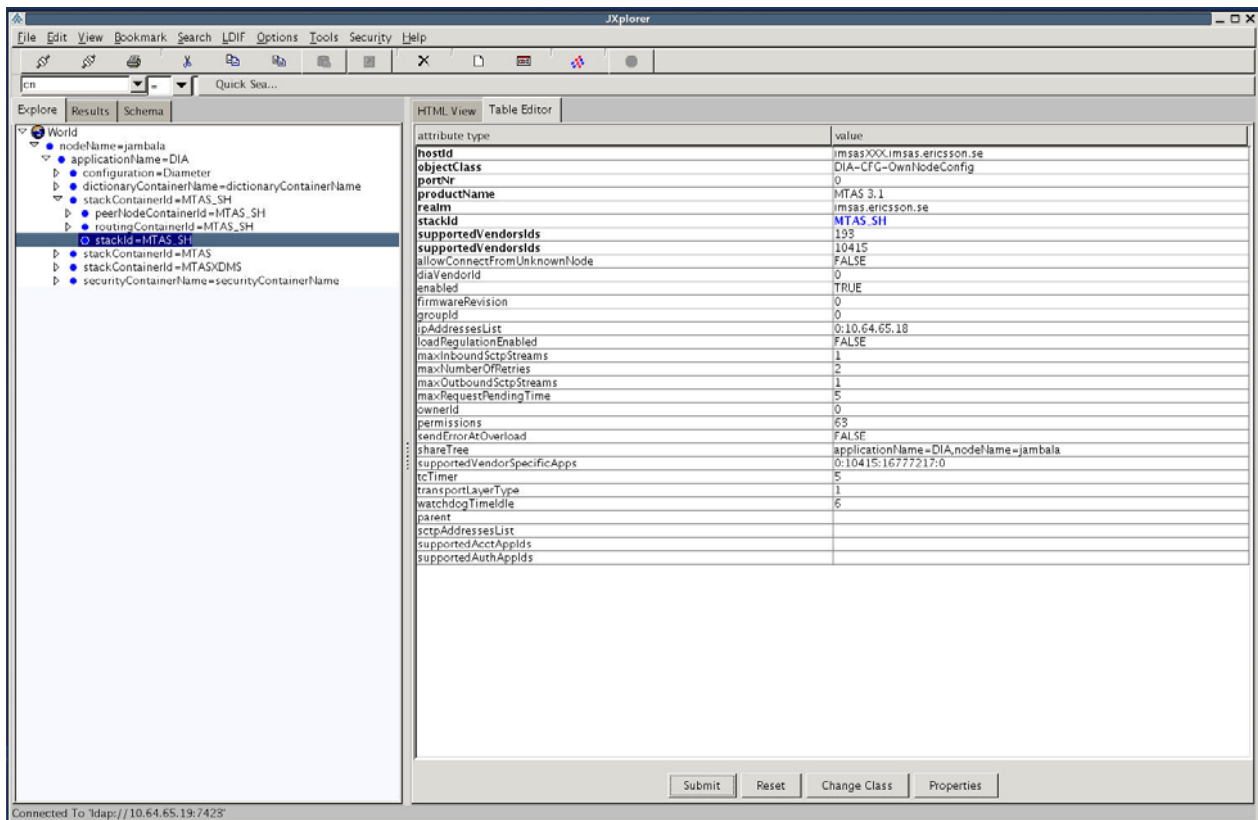


Figure 10 CM Browser Snapshot Example, Diameter Stack

See Figure 11 for a browser view example of Diameter stack attributes for Realm Routing Table Based Request Routing.

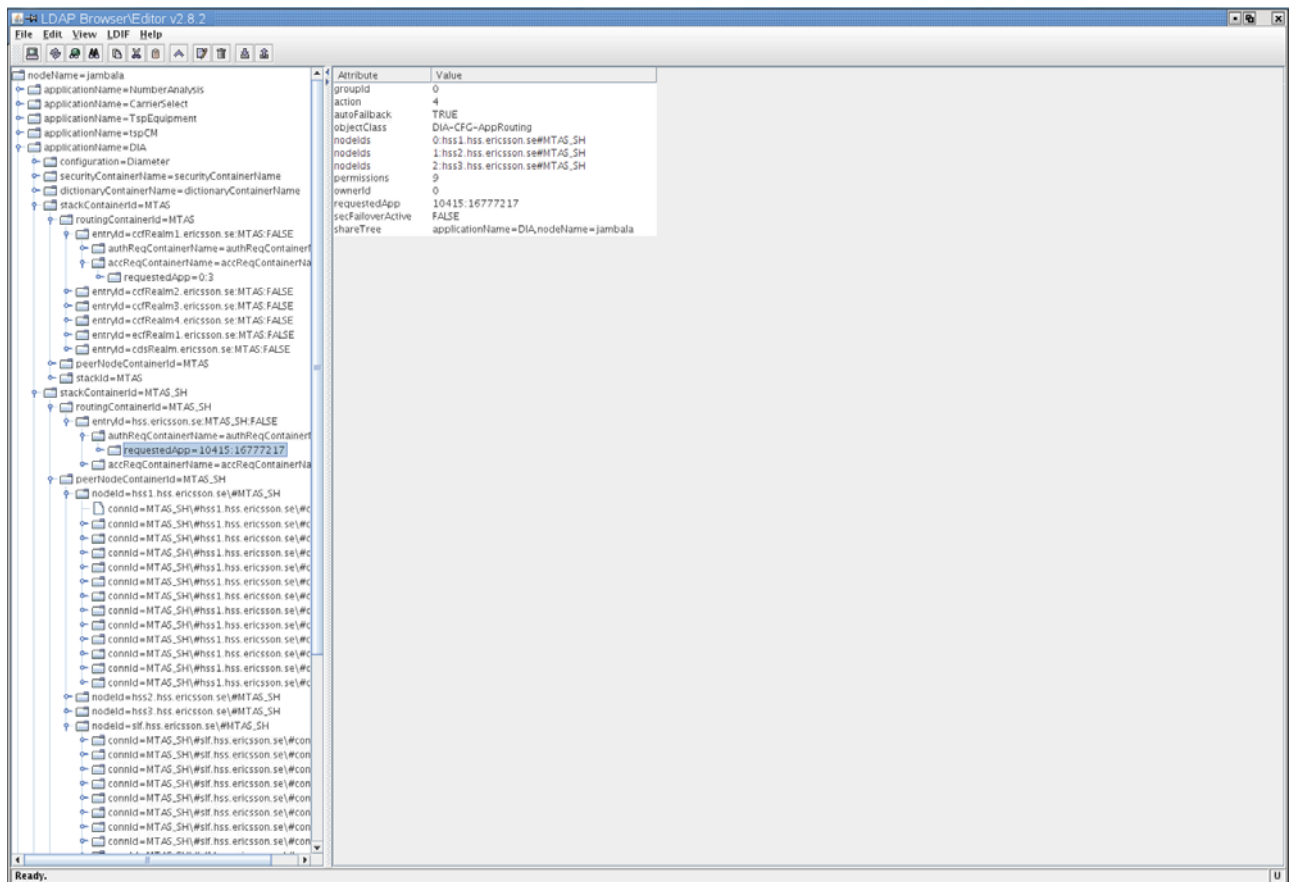


Figure 11 CM Browser Snapshot Example, Diameter Stack, Realm Routing Table Based Request Routing

3.1 Diameter Stack Configuration Attributes

All attributes in Table 2, Table 3, and Table 6 must be set. If the `mtasShIfDestinationHost` MTAS application-specific configuration attribute is set, then do not set the attribute in Table 5. If the MTAS is configured for Realm Routing Table Based Request Routing (see `mtasShIfRealmBasedRouting` configuration attribute in Section 4 on page 43), then the attribute in Table 5 must be set. The Diameter stack configuration attributes have dependencies to the MTAS application configuration attributes, and are to be used together.

For more information on the LDAP configuration attribute identities, refer to *Managed Object Model (MOM)*.

3.1.1 Own Node Configuration for Subscriber Data/XDMS Functions

The values of the `DIA-CFG-OwnNodeConfig` MO for Subscriber Data/XDMS functions are defined in Table 2 and in *Managed Object Model (MOM)*.



Table 2 MTAS Attributes for Own Node Diameter Configuration (DIA-CFG-OwnNodeConfig)

Attribute	Description	Value
stackId	The unique name of the stack instance. One <code>stackId</code> is used for the Subscriber Data function, and another <code>stackId</code> for the XDMS function. Each stack instance has its own set of Diameter configuration attribute values.	For the MTAS_SH stack: <ul style="list-style-type: none">• MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">• MTASXDMS
productName	The node product name.	ericsson_mtas
supportedVendorSpecificApps	List of the Diameter applications that supports Sh requests.	0:10415:16777217:0
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
hostId	The node identification.	Example: mtas1.mtas.operator.com
realm	The domain to which the node must belong.	Example: mtas.operator.com
ipAddressesList	The IP address that identifies the own node.	Example: IPv4 - 0:130.100.209.130 IPv6 - 0:x:x:x:x:x:x:x The x is a hexadecimal value of a 16-bit piece of the address.
watchdogTimeIdle	The timer value between watchdog messages on an idle link.	6
maxNumberOfRetries	This attribute is the maximum number of times the system retries to send a request.	1
maxRequestPendingTime	This attribute is the maximum time without receiving a response for a request.	3



Table 2 MTAS Attributes for Own Node Diameter Configuration (DIA-CFG-OwnNodeConfig)

Attribute	Description	Value
tcTimer	This attribute is the time elapsed between reattempts when the connection to a node has failed.	3
portNr	<p>The port number that the Diameter stack uses.</p> <p>One portNr is used for the Subscriber Data function, and another portNr for the XDMS function.</p>	<p>Example:</p> <p>For the MTAS_SH stack:</p> <ul style="list-style-type: none"> • MTAS_SH - 3868 <p>For the MTASXDMS stack:</p> <ul style="list-style-type: none"> • MTASXDMS - 3869

Each node in the realm configures its own DIA-CFG-OwnNodeConfig MO.

Note: Because the Subscriber Data/XDMS functions act as clients, the watchdogTimeIdle, maxNumberOfRetries, and maxRequestPendingTime attributes must not be configured on the other peer nodes. However, if they are configured, then they must be configured to the same values as here.

3.1.2

Neighbor Node Data Configuration for Subscriber Data/XDMS Functions

Data for other Diameter peer nodes in the network must be configured for Subscriber Data/XDMS functions. This makes it possible for the Subscriber Data/XDMS function to set up communication with those nodes. One DIA-CFG-NeighbourNode object is created for each peer node in the network. If the HSS is deployed in data layered network architecture, then only the DRA nodes need to be configured.

The values of the DIA-CFG-NeighbourNode MO for the Subscriber Data/XDMS functions are defined in Table 3. For more information, refer to *Managed Object Model (MOM)*.

Table 3 MTAS Attributes for Neighbor Node Configuration (*DIA-CFG-NeighbourNode*)

Attribute	Description	Value
nodeId	The identifier of the node. Composed of hostId and stackId. It is a read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">hss1.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">hss1.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	Example: 0:10.0.194.130
enabled	Enables or disables the connection.	TRUE or FALSE

When a neighbor node is added to the stack, it contains one connection (*DIA-CFG-Conn*) labeled `conn1` by default. Multiple connections can be added (it is recommended that one connection is defined per Dicos Traffic Processor in the MTAS node), if supported by the peer node (SLF, HSS, or DRA). Expansion to multiple connections is supported by the Ericsson SLF, HSS, and DRA, but not necessarily by other vendors since it is not included in the Diameter standard. The value of the *DIA-CFG-Conn* MO for Subscriber Data/XDMS functions is defined in Table 4.



Table 4 MTAS Attributes for Neighbor Node Connections (DIA-CFG-Conn)

Attribute	Description	Value
connId	The identifier of the connection. Composed of stackId, hostId, and connId. It must be unique among the connections for this neighbor node, for example, conn1, conn2. It is a read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none"> MTAS_SH#hss1.hss.operator.com#conn1 For the MTASXDMS stack: <ul style="list-style-type: none"> MTASXDMS#hss1.hss.operator.com#conn1
enabled	Enables or disables the connection.	TRUE or FALSE

3.1.3 Add Realm Routing Table Data

The most important settings of the RRT are defined in Table 5 and Table 6. The Realm-related configuration is set through the following MOs:

- *DIA-CFG-Drt*, see Table 5
- *DIA-CFG-AppRouting*, see Table 6

For information on how to configure the MOs, refer to *Managed Object Model (MOM)*.

Table 5 MTAS Attributes for Realm Routing Table Configuration (DIA-CFG-Drt)

Attribute	Description	Value
entryId	The entryId consists of realm, stackId, and isIncomingRequest. The isIncomingRequest field in the entryId attribute is true for routing incoming requests, and false for outgoing requests.	For the MTAS_SH stack: <ul style="list-style-type: none"> hss.operator.com:MTAS_SH: FALSE For the MTASXDMS stack: <ul style="list-style-type: none"> hss.operator.com:MTASXDMS: FALSE

The mandatory entryId attribute has three parts:

- The realm of the client
- The stack instance to identify the Own Node



- The indication of whether this entry of the RRT is used to route incoming or outgoing requests

In Table 5, the Subscriber Data/XDMS functions are clients and send requests to the SLF, HSS, and DRA, which act as Diameter servers. The `isIncomingRequest` indicator is set to `FALSE` for the Subscriber Data/XDMS functions. The mandatory action field must be set to none, indicating that the Subscriber Data/XDMS functions act as a client.

See Table 6 for information on setting the Authorization Application Routing attributes for Subscriber Data and XDMS functions.

Table 6 Authorization Application Routing Configuration for Subscriber Data and XDMS Functions (*DIA-CFG-AppRouting*)

Attribute	Description	Value
<code>requestedApp</code>	The vendor Diameter application whose messages are recognized by the RRT.	10415:16777217
<code>action</code>	The routing action from requests for a certain realm and a given request type that belongs to the Diameter application specified in the <code>requestedApp</code> attribute.	4 ⁽¹⁾
<code>nodeIds</code>	One or more servers that the message is to be routed to.	For the MTAS_SH stack: <ul style="list-style-type: none"> • 0:<neighbor node id>#MTAS_SH 0:hss1.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none"> • 0:<neighbor node id>#MTASXDMS 0:hss1.hss.operator.com#MTASXDMS

(1) No action is taken. The Subscriber Data/XDMS functions can only have the value 4 because the `isIncomingRequest` field of the `entryId` attribute is set to `false` in *DIA-CFG-Drt*.

If the HSS is deployed in a data layered network architecture, then add the `nodeId` of each DRA node from the Neighbor Node Configuration (see Table 3) to the `nodeIds` attribute in the *DIA-CFG-AppRouting* MO.

3.2 Diameter Stack Configuration Examples

These examples show configurations for the different HSS network architectures described in Section 2.3.1 Supported HSS Network Architectures on page 6.



3.2.1 Example Diameter Stack Configuration for Classic HSS Network Architecture with a Single Monolithic HSS Instance

The following is assumed:

- One HSS instance is accessible, in this example `hss1.hss.operator.com` is used, with two configured IP addresses `10.0.194.130` and `10.0.194.131` on TCP port `3872`.
- Two *DIA-CFG-NeighbourNode* MO instances are created, one for the MTAS_SH stack and one for the MTASXDMS stack.

The value of the *DIA-CFG-NeighbourNode* MO for the HSS1 peer-node is defined in Table 7.

Table 7 Attributes for Peer-Node HSS1

Attribute	Description	Value
<code>nodeId</code>	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none"> • <code>hss1.hss.operator.com#MTAS_SH</code> For the MTASXDMS stack: <ul style="list-style-type: none"> • <code>hss1.hss.operator.com#MTASXDMS</code>
<code>initiateConnection</code>	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
<code>transportLayerType</code>	The transport protocol used when setting up a connection to the node.	1 (TCP)
<code>ipAddressesList</code>	The IP address that identifies the neighbor node.	0:10.0.194.130 1:10.0.194.131
<code>portNr</code>		3872
<code>enabled</code>	Enables or disables the connection.	TRUE

Set the attributes as follows:

- The *DIA-CFG-Drt* MO (and its descendants: *DIA-CFG-AuthReqContainer*, *DIA-CFG-AppRouting*) is left empty.
- The `mtasShIfDestinationHost` MTAS application configuration attribute is set to `hss1.hss.operator.com`.



- The `mtasShIfDestinationRealm` attribute is set to `hss.operator.com`.
- The `mtasShIfRealmBasedRouting` attribute is set to 0.

For how to configure the Sh interface, see Section 4 on page 43.

3.2.2 Example Diameter Stack Configuration for Classic HSS Network Architecture with Multiple SLFs in REDIRECT Mode and Multiple HSS Instances

The following is assumed:

- Two HSS instances are accessible, in this example:
 - HSS1: `hss1.hss.operator.com` with two configured IP addresses 10.0.194.130 and 10.0.194.131 on TCP port 3872.
 - HSS2: `hss2.hss.operator.com` with two configured IP addresses 10.0.194.140 and 10.0.194.141 on TCP port 3873.
- Two SLF instances are accessible, in this example:
 - SLF1: `slf1.hss.operator.com` with two configured IP addresses 10.0.194.150 and 10.0.194.151 on TCP port 3872.
 - SLF2: `slf2.hss.operator.com` with two configured IP addresses 10.0.194.160 and 10.0.194.161 on TCP port 3873.

Settings for DIA-CFG-NeighbourNode MO Instances

The value of the *DIA-CFG-NeighbourNode* MO for the HSS1, HSS2, SLF1, and SLF2 peer-nodes are defined in Table 8 through Table 11.

Note: Two *DIA-CFG-NeighbourNode* MO instances must be created for all HSS/SLF peer nodes, one for the `MTAS_SH` stack and one for the `MTASXDMS` stack.

Table 8 Attributes for Peer-Node HSS1

Attribute	Description	Value
<code>nodeId</code>	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the <code>MTAS_SH</code> stack: <ul style="list-style-type: none">• <code>hss1.hss.operator.com#MTAS_SH</code> For the <code>MTASXDMS</code> stack: <ul style="list-style-type: none">• <code>hss1.hss.operator.com#MTASXDMS</code>



Table 8 Attributes for Peer-Node HSS1

Attribute	Description	Value
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.130 1:10.0.194.131
portNr		3872
enabled	Enables or disables the connection.	TRUE

Table 9 Attributes for Peer-Node HSS2

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: • hss2.hss.operator.com#MTAS_SH For the MTASXDMS stack: • hss2.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.140 1:10.0.194.141
portNr		3873
enabled	Enables or disables the connection.	TRUE



Table 10 Attributes for Peer-Node SLF1

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">• <code>slf1.hss.operator.com#MTAS_SH</code> For the MTASXDMS stack: <ul style="list-style-type: none">• <code>slf1.hss.operator.com#MTASXDMS</code>
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.150 1:10.0.194.151
portNr		3872
enabled	Enables or disables the connection.	TRUE

Table 11 Attributes for Peer-Node SLF2

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">• <code>slf2.hss.operator.com#MTAS_SH</code> For the MTASXDMS stack: <ul style="list-style-type: none">• <code>slf2.hss.operator.com#MTASXDMS</code>
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)



Table 11 Attributes for Peer-Node SLF2

Attribute	Description	Value
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.160 1:10.0.194.161
portNr		3873
enabled	Enables or disables the connection.	TRUE

Settings for DIA-CFG-Drt MO Instances

Settings of the *DIA-CFG-Drt* MO for realm `hss.operator.com` are described in Table 12.

Note: Two *DIA-CFG-Drt* MO instances must be created, one for the `MTAS_SH` stack and one for the `MTASXDMS` stack.

Table 12 Configuration Attributes for Diameter Routing Table *DIA-CFG-Drt* for Realm *hss.operator.com*

Attribute	Description	Value
entryId	The <code>entryId</code> consists of <code>realm</code> , <code>stackId</code> , and <code>isIncomingRequest</code> . The <code>isIncomingRequest</code> field in the <code>entryId</code> attribute is true for routing incoming requests, and false for outgoing requests.	Example: For the <code>MTAS_SH</code> stack: <ul style="list-style-type: none"> <code>hss.operator.com#MTAS_SH</code> FALSE For the <code>MTASXDMS</code> stack: <ul style="list-style-type: none"> <code>hss.operator.com#MTASXDMS</code> FALSE

Settings for DIA-CFG-AppRouting under Configured DIA-CFG-Drt Instance

Settings of the *DIA-CFG-AppRouting* MO instance under the configured *DIA-CFG-Drt* MO instance are described in Table 13.

Note: Two *DIA-CFG-AppRouting* MO instances must be created, one for the `MTAS_SH` stack and one for the `MTASXDMS` stack.



Table 13 Configuration Attributes for Diameter Routing Table DIA-CFG-AppRouting under DIA-CFG-Drt Routing Table

Attribute	Description	Value
requestedApp	The vendor Diameter application whose messages are recognized by the RRT.	10415:16777217
action	The routing action from requests for a certain realm and a given request type that belongs to the Diameter application specified in the requestedApp attribute.	4
nodeIds	One or more servers that the message is to be routed to.	For the MTAS_SH stack: <ul style="list-style-type: none">• 0:slf1.hss.operator.com#MTAS_SH1:slf2.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">• 0:slf1.hss.operator.com#MTASXDMS1:slf2.hss.operator.com#MTASXDMS

Set the attributes as follows:

- The `mtasShIfDestinationHost` MTAS application configuration attribute is left empty.
- The `mtasShIfDestinationRealm` attribute is set to `hss.operator.com`.
- The `mtasShIfRealmBasedRouting` attribute is set to 0.

For how to configure the Sh interface, see Section 4 on page 43.

3.2.3 Example Diameter Stack Configuration for Classic HSS Network Architecture with Multiple SLFs in PROXY Mode and Multiple HSS Instances

The following is assumed:

- Two HSS instances are accessible, in this example:
 - HSS1: `hss1.hss.operator.com` with two configured IP addresses `10.0.194.130` and `10.0.194.131` on TCP port 3872.



- HSS2: `hss2.hss.operator.com` with two configured IP addresses `10.0.194.140` and `10.0.194.141` on TCP port `3873`.
- Two SLF instances are accessible, in this example:
 - SLF1: `slf1.hss.operator.com` with two configured IP addresses `10.0.194.150` and `10.0.194.151`.
 - SLF2: `slf2.hss.operator.com` with two configured IP addresses `10.0.194.160` and `10.0.194.161`.
 - SLF1 and SLF2 are both accessible on TCP port `3872`.

Settings for DIA-CFG-NeighbourNode MO Instances

The value of the *DIA-CFG-NeighbourNode* MO for the HSS1, HSS2, SLF1, and SLF2 peer-nodes are defined in Table 14 through Table 17.

Note: Two *DIA-CFG-NeighbourNode* MO instances must be created for all HSS/SLF peer nodes, one for the `MTAS_SH` stack and one for the `MTASXDMS` stack.

Table 14 Attributes for Peer-Node HSS1

Attribute	Description	Value
<code>nodeId</code>	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the <code>MTAS_SH</code> stack: <ul style="list-style-type: none"> • <code>hss1.hss.operator.com#MTAS_SH</code> For the <code>MTASXDMS</code> stack: <ul style="list-style-type: none"> • <code>hss1.hss.operator.com#MTASXDMS</code>
<code>initiateConnection</code>	This attribute is set to <code>TRUE</code> when the Diameter Node initiates a connection with the neighbor node.	<code>TRUE</code>
<code>transportLayerType</code>	The transport protocol used when setting up a connection to the node.	<code>1 (TCP)</code>



Table 14 Attributes for Peer-Node HSS1

Attribute	Description	Value
ipAddressesList	The IP address that identifies the neighbor node.	For the Classic SLF PROXY architecture, SLF addresses are configured for the HSS peer-nodes: 0:10.0.194.150 1:10.0.194.151 2:10.0.194.160 3:10.0.194.161
portNr		3872
enabled	Enables or disables the connection.	TRUE

Table 15 Attributes for Peer-Node HSS2

Attribute	Description	Value
nodeId	The identifier of the node. Composed of hostId and stackId. Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">hss2.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">hss2.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)



Table 15 Attributes for Peer-Node HSS2

Attribute	Description	Value
ipAddressesList	The IP address that identifies the neighbor node.	For the Classic SLF PROXY architecture, SLF addresses are configured for the HSS peer-nodes: 0:10.0.194.150 1:10.0.194.151 2:10.0.194.160 3:10.0.194.161
portNr		3872
enabled	Enables or disables the connection.	TRUE

Table 16 Attributes for Peer-Node SLF1

Attribute	Description	Value
nodeId	The identifier of the node. Composed of hostId and stackId. Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">slf1.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">slf1.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.150 1:10.0.194.151
portNr		3872
enabled	Enables or disables the connection.	TRUE



Table 17 Attributes for Peer-Node SLF2

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">• <code>slf2.hss.operator.com#MTAS_SH</code> For the MTASXDMS stack: <ul style="list-style-type: none">• <code>slf2.hss.operator.com#MTASXDMS</code>
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.160 1:10.0.194.161
portNr		3872
enabled	Enables or disables the connection.	TRUE

Settings for DIA-CFG-Drt MO Instances

Settings of the *DIA-CFG-Drt* MO for realm `hss.operator.com` are defined in Table 18.

Note: Two *DIA-CFG-Drt* MO instances must be created, one for the MTAS_SH stack and one for the MTASXDMS stack.



Table 18 Configuration Attributes for Diameter Routing Table *DIA-CFG-Drt* for Realm *hss.operator.com*

Attribute	Description	Value
entryId	The entryId consists of realm, stackId, and isIncomingRequest. The isIncomingRequest field in the entryId attribute is true for routing incoming requests, and false for outgoing requests.	Example: For the MTAS_SH stack: <ul style="list-style-type: none"> hss.operator.com#MTAS_SH FALSE For the MTASXDMS stack: <ul style="list-style-type: none"> hss.operator.com#MTASXDMS FALSE

Settings for DIA-CFG-AppRouting under Configured DIA-CFG-Drt Instance

Settings of the *DIA-CFG-AppRouting* MO under the configured *DIA-CFG-Drt* MO instance are defined in Table 19.

Note: Two *DIA-CFG-AppRouting* MO instances must be created, one for the MTAS_SH stack and one for the MTASXDMS stack.

Table 19 Configuration Attributes for Diameter Routing Table *DIA-CFG-AppRouting* under *DIA-CFG-Drt* Routing Table

Attribute	Description	Value
requestedApp	The vendor Diameter application whose messages are recognized by the RRT.	10415:16777217



Table 19 Configuration Attributes for Diameter Routing Table DIA-CFG-AppRouting under DIA-CFG-Drt Routing Table

Attribute	Description	Value
action	The routing action from requests for a certain realm and a given request type that belongs to the Diameter application specified in the requestedApp attribute.	4
nodeIds	One or more servers that the message is to be routed to.	For the MTAS_SH stack: <ul style="list-style-type: none">• 0:slf1.hss.operator.com#MTAS_SH1:slf2.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">• 0:slf1.hss.operator.com#MTASXDMS1:slf2.hss.operator.com#MTASXDMS

Set the attributes as follows:

- The `mtasShIfDestinationHost` MTAS application configuration attribute is left empty.
- The `mtasShIfDestinationRealm` attribute is set to `hss.operator.com`.
- The `mtasShIfRealmBasedRouting` attribute is set to 0.

For how to configure the Sh interface, see Section 4 on page 43.

3.2.4 Example Diameter Stack Configuration for HSS-FE Deployment or HSS Data Layered Network Architecture

The following is assumed:

- Two DRA instances are accessible, in this example:
 - DRA1: `dra1.hss.operator.com` with two configured IP addresses `10.0.194.130` and `10.0.194.131` on TCP port 3872.
 - DRA2: `dra2.hss.operator.com` with two configured IP addresses `10.0.194.140` and `10.0.194.141` on TCP port 3873.



Settings for DIA-CFG-NeighbourNode MO Instances

The value of the *DIA-CFG-NeighbourNode* MO for DRA1 and DRA2 peer-nodes are defined in Table 20 and Table 21.

Note: Two *DIA-CFG-NeighbourNode* MO instances must be created for all DRA peer nodes, one for the MTAS_SH stack and one for the MTASXDMS stack.

Table 20 Attributes for Peer-Node DRA1

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">dra1.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">dra1.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.130 1:10.0.194.131
portNr		3872
enabled	Enables or disables the connection.	TRUE



Table 21 Attributes for Peer-Node DRA2

Attribute	Description	Value
nodeId	The identifier of the node. Composed of <code>hostId</code> and <code>stackId</code> . Read-only attribute.	Example: For the MTAS_SH stack: <ul style="list-style-type: none">• dra2.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">• dra2.hss.operator.com#MTASXDMS
initiateConnection	This attribute is set to TRUE when the Diameter Node initiates a connection with the neighbor node.	TRUE
transportLayerType	The transport protocol used when setting up a connection to the node.	1 (TCP)
ipAddressesList	The IP address that identifies the neighbor node.	0:10.0.194.140 1:10.0.194.141
portNr		3873
enabled	Enables or disables the connection.	TRUE

Settings for DIA-CFG-Drt MO Instances

Settings of the *DIA-CFG-Drt* MO for realm `hss.operator.com` are defined in Table 22.

Note: Two *DIA-CFG-Drt* MO instances must be created, one for the MTAS_SH stack and one for the MTASXDMS stack.



Table 22 Configuration Attributes for Diameter Routing Table *DIA-CFG-Drt* for Realm *hss.operator.com*

Attribute	Description	Value
entryId	The entryId consists of realm, stackId, and isIncomingRequest. The isIncomingRequest field in the entryId attribute is true for routing incoming requests, and false for outgoing requests.	Example: For the MTAS_SH stack: <ul style="list-style-type: none"> hss.operator.com#MTAS_SH FALSE For the MTASXDMS stack: <ul style="list-style-type: none"> hss.operator.com#MTASXDMS FALSE

Settings for DIA-CFG-AppRouting under Configured DIA-CFG-Drt Instance

Settings of the *DIA-CFG-AppRouting* MO under the configured *DIA-CFG-Drt* MO instance are defined in Table 23.

Note: Two *DIA-CFG-AppRouting* MO instances must be created, one for the MTAS_SH stack and one for the MTASXDMS stack.

Table 23 Configuration Attributes for Diameter Routing Table *DIA-CFG-AppRouting* under *DIA-CFG-Drt* Routing Table

Attribute	Description	Value
requestedApp	The vendor Diameter application whose messages are recognized by the RRT.	10415:16777217



Table 23 Configuration Attributes for Diameter Routing Table DIA-CFG-AppRouting under DIA-CFG-Drt Routing Table

Attribute	Description	Value
action	The routing action from requests for a certain realm and a given request type that belongs to the Diameter application specified in the requestedApp attribute.	4
nodeIds	One or more servers that the message is to be routed to.	For the MTAS_SH stack: <ul style="list-style-type: none">• 0:dra1.hss.operator.com#MTAS_SH1:dra2.hss.operator.com#MTAS_SH For the MTASXDMS stack: <ul style="list-style-type: none">• 0:dra1.hss.operator.com#MTASXDMS1:dra2.hss.operator.com#MTASXDMS

Set the attributes as follows:

- The `mtasShIfDestinationHost` MTAS application configuration attribute is left empty.
- The `mtasShIfDestinationRealm` attribute is set to `hss.operator.com`.
- The `mtasShIfRealmBasedRouting` attribute is set to 1.

For how to configure the Sh interface, see Section 4 on page 43.



4 Configure Sh Interface

To route Sh messages correctly, it is necessary to specify some MTAS application-specific configuration attributes besides the Diameter stack configuration attributes. The Sh configuration attributes of the Subscriber Data function are shared with the XDMS function.

To configure the Sh interface on the MTAS application side:

1. Navigate to the *MtasShIf* MO.
2. Set the `mtasShIfDestinationRealm` attribute value to the realm in which the HSS, SLF, or DRA nodes reside, for example, `hss.operator.com`.

This value is used as the value of the Destination-Realm AVP in all Sh requests sent by MTAS.

Note: If the `mtasSubsDataRegistrationMode` is set to 2 (No Register Mode) in the `mtasSubsData` MO, then step 3 and step 4 are not needed.

3. Use one of the following options for the `mtasShIfDestinationHost` attribute:
 - If there is more than one HSS present in the network, then leave the `mtasShIfDestinationHost` attribute empty.
 - If the HSS is deployed in data layered architecture, then leave the `mtasShIfDestinationHost` attribute empty.
 - Otherwise, set the `mtasShIfDestinationHost` attribute to the HSS hostname, for example, `hss1.hss.operator.com`.
4. Set the `mtasShIfMmtelServiceInd` attribute to configure the MMTel service data indication. The attribute value must match the service indication used in the HSS to store the MMTel service data, for example, `MmtServiceConfig`.
5. Set the `mtasShIfMmtelGroupServiceInd` attribute to configure the MMTel group service data indication. The attribute value must match the service indication used in the HSS to store the MMTel group service data, for example, `MmtGroupServiceConfig`.
6. Set the `mtasShIfMmtelServiceProfileInd` attribute to configure the MMTel Service Profile data indication. The attribute value must match the service indication used in the HSS to store the MMTel Service Profile data, for example `MmtServiceProfileConfig`.



7. If the SIP Trunking function is to be used, set the `mtasShIfStReferralsServiceInd` attribute to configure the SIP Trunking Referral data indication. The attribute value must match the service indication used in the HSS to store the SIP Trunking referral data, for example, `StReferralConfig`.
8. If the SIP Trunking function is to be used, set the `mtasShIfStServiceDataInd` attribute to configure the SIP Trunking Service Data indication. The attribute value must match the service indication used in the HSS to store the SIP Trunking Service Data, for example, `StServiceConfig`.
9. If the Sh interface efficiency function is to be used, set the `mtasShIfEfficiency` attribute to 1. If the `mtasShIfEfficiency` is set, there is the possibility to change the default behavior of the feature with two additional attributes. The default values of these attributes are set to standard mode. The Notif-Eff capability is always propagated in all Sh messages.

The `mtasShIfEffDiscoveryMode` attribute specifies how the MTAS propagates its Update-Eff capability in the Sh messages sent towards the HSS. The `mtasShIfEffDiscoveryMode` can be set to standard or intensive mode.

The `mtasShIfEffMandatoryBitSetting` attribute specifies how the MTAS sets the Mandatory bit (M bit) in the Supported-Features AVP header of the Sh messages. The `mtasShIfEffMandatoryBitSetting` attribute can be set to standard or informative mode.

10. If the Realm Routing Table Based Request Routing is to be used, set `mtasShIfRealmBasedRouting` attribute to 1, otherwise leave it as 0.

Note: If the `mtasShIfRealmBasedRouting` attribute is set to 1, the `mtasShIfDestinationHost`, `mtasShIfEfficiency` attributes, and related settings are ignored.

11. Click **Submit**.
 12. Perform a backup. For more information, refer to *Create Backup*.
- See Figure 12 for a browser view example of Sh configuration attributes.

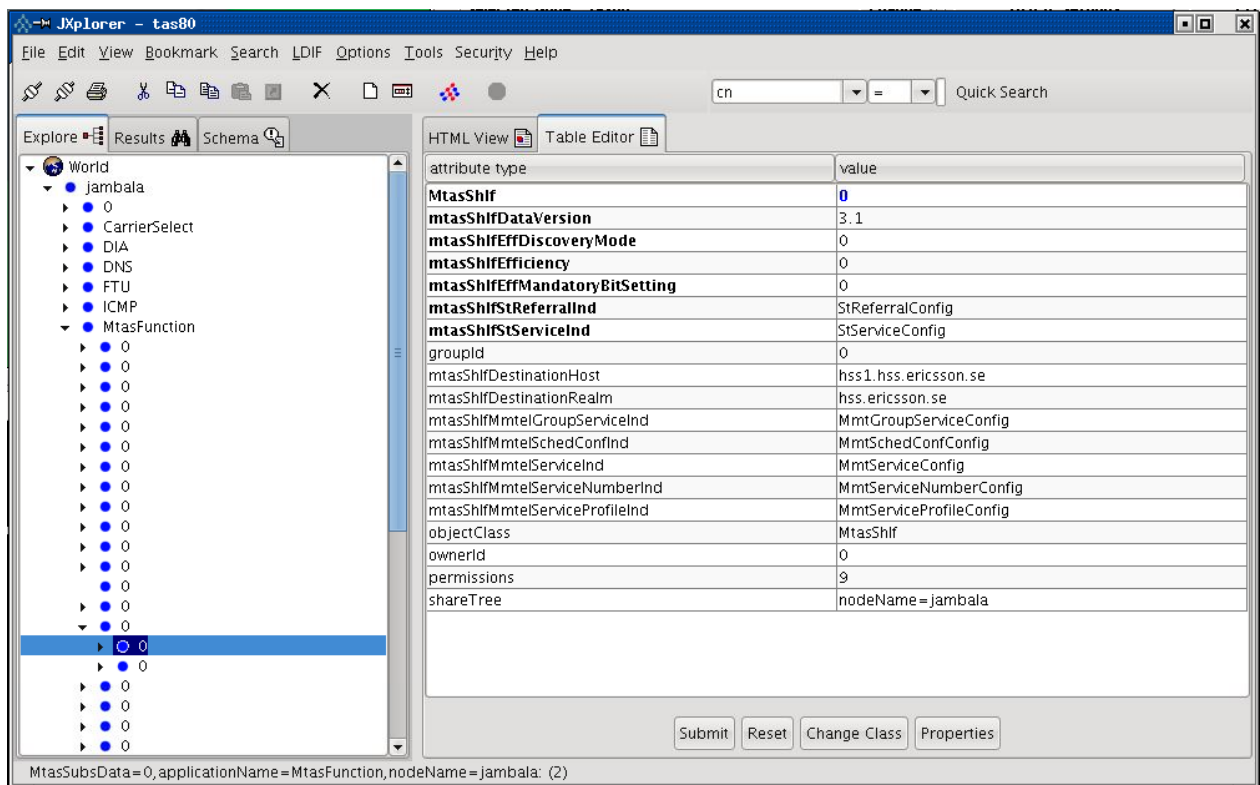


Figure 12 CM Browser Snapshot Example, MtasShlf MO Attributes





5 Caching Contact Data and UE Terminal Type Classification Configuration

Caching contact data is enabled by setting `mtasSubsDataCacheContactData` attribute to 1.

Caching contact data is set to `enabled` by default. Disable it if not needed, to avoid redundant traffic and increase performance.

When CM attribute `mtasSubsDataCachePani` is set to 1 = enabled, P-Access-
-Network-Info (PANI) from SIP REGISTER/INVITE/180/183/200(INVITE) is also saved in the Contact Data. If `mtasSubsDataCachePani` is set to 2 = enabled for registration only, PANI is saved to Contact Data only from REGISTER, Notify, and Initial INVITE on unregistered users in originating MMTel AS. The CCR-I message contains PANI from the listed messages, and terminating MTAS sends CCR-U message with PANI from 180/183/200OK messages. CM attribute `mtasSubsDataCacheContactData` must be enabled before enabling `mtasSubsDataCachePani`.

Caching contact data is required specifically by IMS Centralized Services (ICS) on SCC AS and Flexible Communication Distribution (FCD) to Primary User's Devices. For more information about the ICS on SCC AS, refer to *MTAS IMS Centralized Services Management Guide*. For more information about the FCD to Primary User's Devices, refer to *MTAS Flexible Communication Distribution Management Guide*.

Subscriber Data function caches registered contacts information, parsed from the body of third-party REGISTER message from S-CSCF (SIP REGISTER).

For this purpose, the iFC in HSS is configured to trigger initial registration, re-registration, and deregistration to the MTAS and to include the user REGISTER request and 200 OK response in the "message/sip" body of the third-party registration.

If the administrative state for SCC AS is unlocked, the HSS-based ATCF info storage is enabled (`mtasSubsDataSccAtcfInfoInHss` attribute set to 1) and the third-party REGISTER request contains ATCF registration information of the registered user, this information is stored in HSS as transparent data.

The MTAS supports two methods to provide ATCF registration information to the SCC AS registration procedure. In the Ericsson proprietary extension (Feature-Caps header and IMPI), ATCF information is provided with one-time subscription for reg-event from CSCF. The ATCF info storage is an alternative way to provide ATCF registration information by storing ATCF information in the HSS as transparent data through the Sh interface.

The SCC AS registration procedure can be configured in the MTAS on node level to use the one-time subscription for reg-event or the ATCF info storage.

If caching is enabled, but the Contact Data cannot be fetched from the body of third-party REGISTER for some reason (for example, the body or a part of it is missing or broken), MTAS behaves differently, depending on deployment configuration.

- MMTel AS standalone tries to retrieve the Contact Data from S-CSCF using one time reg-event subscription.
- SCC AS (standalone or co-located with MMTel AS) rejects the third-party REGISTER with a SIP 400 Bad Request response with the Warning header set to 399: "MIME body message/sip content not sufficient".

For more information about the MMTel AS registration procedures, refer to *MTAS External Network Configuration*.

If caching is enabled, but the Contact Data is not locally cached on MTAS when a call establishment is made (when originating or terminating initial INVITE is received), for example in failover scenarios, Contact Data is fetched from S-CSCF with explicit one-time reg-event subscription and cached.

It is possible to configure MTAS to use its FQDN identity in From and P-Asserted-Identity headers in the SUBSCRIBE message for reg-event packages toward I/S-CSCF instead of its IP address.

When contact data is being retrieved from S-CSCF with explicit one time reg-event subscription, `mtasSubsDataRegEventResponseTimer` setting defines the time that MTAS waits for a response to SUBSCRIBE sent to the S-CSCF to obtain the registration status of a served user. The same setting also defines the time that MTAS waits for a NOTIFY after receipt of a 2xx response to the SUBSCRIBE. If the S-CSCF supports the Ericsson proprietary extension (Feature-Caps header and IMPI), the ATCF registration information required for SCC AS used with SRVCC Release 10 is present in the NOTIFY. If not, and the `mtasSubsDataScAtcfInfoInHss` attribute is enabled, then the ATCF information is retrieved from HSS. For the ATCF registration information to be available in the HSS, the HSS-based ATCF info storage must have been enabled at registration of the user. Otherwise, the user must be deregistered and then registered for the change to take effect, for example if the ATCF info storage is enabled at upgrade.

If IMPI has not been parsed from the reg-event NOTIFY and is not stored in MTAS, like in 3GPP R9, the IMPI is updated based on the P-Com.PrivateUserID header, if available. For the originating session case P-Com.PrivateUserID header from the INVITE is used, and for terminating session case P-Com.PrivateUserID header from 200 OK response to INVITE is used.

Contact data remains in cache until expiry of registration timer, which set in `mtasSubsDataDefaultRegTimer` CM attribute. The default value of the registration timer is 21600 minutes.

If the value of `mtasSubsDataRegEventNotifier` is 1 (I-CSCF), MTAS fetches the Contact Data from I-CSCF using one time reg-event subscription.



UE terminal type classification (mobile/VoLTE or fixed) during 3rd-party registration and processing of reg-event notification for caching Contact Data purpose can be based on either the feature tags of a registered user, or the PANI header. Feature tags, being basis for device mobility classification as mobile/VoLTE, are configurable by attribute `mtasSubsDataMobileClassification`. If it contains at least one entry, the classification is based on feature tags listed in this CM attribute only, without taking P-Access-Network-Info header into account. Otherwise, if this setting is empty (default value), a device is classified as “mobile” based on the P-Access-Network-Info header indicating 3GPP GERAN, 3GPP UTRAN or 3GPP E-UTRAN access, and if P-Access-Network-Info header is absent – based on presence of `+g.3gpp.ics=“server”` or `+g.3gpp.accesstype=“cellular”` feature tag. If none of the these conditions are fulfilled, a device is classified as “fixed” by default.





6 Rebalancing

The rebalancing feature in MTAS enables the operator to move registered subscribers from a source MTAS node to a target MTAS node after node expansion or after MTAS node recovery.

6.1 Activation

The rebalancing feature is activated manually by the operator after setting values for CM attributes including the threshold and the target node SIP URI. When the feature is active, INITIAL and re-registrations of a subscriber on the source MTAS node are responded with a 305 (Use Proxy) message that contains the address of the target MTAS node in the Contact header. The subscriber is redirected to the target MTAS node and deregistered from the source MTAS node.

6.2 Deactivation

The rebalancing feature can be deactivated manually by operator or automatically when the active number of subscribers registered on the node falls below the configured threshold.

The rebalancing can also be deactivated when subscribers are deregistered from the node upon expiry of the MTAS registration timers `mtasSubsDataDefaultRegTimer` and `mtasSubsDataDeregTimer`.

The values of following MTAS registration timers that are related to the caching interval of user data must be synchronized between the S-CSCF and the MTAS nodes:

mtasSubsDataDefaultRegTimer

This attribute defines what value is used for registration timer for registered users in Subscriber Data component. This timer defines the maximum time subscriber data remains in the cache. The value is to be greater than or equal to the typical registration lifetime in the S-CSCF to allow receiving a reregistration before the expiry of the timer.



mtasSubsDataDeregTimer

This attribute defines the duration of the deregistration timer for unregistered users in Subscriber Data component. This timer defines how long subscriber data remains in the cache after termination of the last session for an unregistered subscriber. The timer is started when the last call for the unregistered subscriber is completed. It is stopped when a new session is initiated for the subscriber.

When rebalancing is active and the registration timer expires for an idle user, the deregistration procedure is implemented as follows:

- If the number of registered users falls below the threshold upon de-registration, the rebalancing feature is deactivated. Else, it remains active.
- The deregistration is graceful. MTAS sends terminating NOTIFY to devices with DEN subscription.



7 Wholesale for Subscriber Data Configuration

The MTAS Subscriber Data only partially supports Wholesale. The `MtasSubsData` MO is the only one of the MOs that control the Subscriber Data function that supports Wholesale. The MO is configurable on Virtual Telephony Provider (VTP) level, that is the VTP operators can have their own configuration in the corresponding `VtasSubsData` MO instances.

Note: The functions to purge and query subscriber data do not support Wholesale.

To configure Wholesale for the MTAS Subscriber Data function, attributes in the `VtasSubsData` MO must be configured and the `vtasSubsDataDropBack` attribute must be set to 0 (use own VTP values).

For more information about the attributes in the `VtasSubsData` MO, refer to *Managed Object Model (MOM)*.

For more information about the Wholesale service, refer to *MTAS Wholesale Support Management Guide*.





8 Examples, Subscriber Data Management

This section describes some subscriber data and some query or purge examples.

8.1 Subscriber Data

A subscriber has a main identity and can have up to seven extra alias identities defined. The subscriber data is kept in the HSS for the main identity and the same data is used for the alias identities.

The query or purge examples from Section 8.2.1 Query or Purge Using Single PUI on page 56 to Section 8.2.6 Query or Purge Using Multiple Wildcards on page 64 are based on the PUI data defined in Table 24.

Table 24 Subscriber Data, Example Data

Default PUI/PUI	Alias/Implicit Identities
sip:+46123456789@someplace.com	tel:+46123456789, sip:+name1@someotherplace.se
sip:+46987654321@someplace2.com	tel:+46987654321, sip:name2@someplace.eu
sip:+44123456543@someplace3.com	tel:+44123456543, sip:name3@someplace.us
sip:+32234567891@someplace4.com	tel:+32234567891
sip:+21345234561@someplace5.co.uk	tel:+21345234561
sip:+3423456789@someplace.org	tel:+3423456789
sip:+2434523456@somewhere.net	
sip:+44gandolf@someotherplace.se	
sip:+pippin@someplace.eu	
sip:samwisegamgee@someplace.us	
sip:777legalos777@someplace.org	



Default PUI/PUI	Alias/Implicit Identities
sip:4souron456@somewhere.net	
sip:243452Gandolf@somewhere.net	

8.2 Query or Purge Examples

This section describes some query or purge examples.

8.2.1 Query or Purge Using Single PUI

An example of a query using Single PUI, sip:+46123456789@someplace.com or tel:+46123456789, is shown in Example 1.

Example 1 Single PUI Query

1. Navigate to the MtasSubsDataMgmt MO.
2. Execute MO action mtasSubsDataMgmtRunQuery with mtasSubsDataMgmtRunQueryPui parameter value sip:+46123456789@someplace.com.

Results

The output file contains the following information:

```
query, 2009-01-10, 13:34:39.974, sip:+46123456789@someplace.com
<alias pui>          <default pui>
tel:+46123456789,    sip:+46123456789@so  ongoing call (N)
                      meplace.com,
sip:name1@someot     sip:+46123456789@so  ongoing call (N)
herplace.se,         meplace.com,
processing           2
completed,
```

8.2.2 Query or Purge Using Wildcard

An example of doing a query or purge using wildcard sip:*xxxxxx or tel:*xxxxxx, is shown in Example 2.

Example 2 Operation, Query, or Purge Using Wildcard

1. Navigate to the MtasSubsDataMgmt MO.



2. Execute MO action `mtasSubsDataMgmtRunQuery` with `mtasSubsDataMgmtRunQueryPui` parameter value `sip:*xxxxxx` or `tel:*xxxxxx`

or execute MO action `mtasSubsDataMgmtRunPurge` with `mtasSubsDataMgmtRunPurgePui` parameter value `sip:*xxxxxx` or `tel:*xxxxxx`.

Results

The output files contain the following information:

query, 2009-01-10, 13:34:39.974, sip:*xxxxxx or tel:*xxxxxx

```
<alias pui>                <default pui>
tel:+3423456789,           sip:+3423456789@som   ongoing call (N)
                           eplace.org,
processing completed,      1
```

purge, 2009-01-10, 13:34:39.974, sip:*xxxxxx or tel:*xxxxxx

```
<default pui>                <alias pui>
sip:+3423456789@some tel:+3423456789
place.org,
sip:777legalos777@som
eplace.org
processing completed,      2
```

8.2.3 Query or Purge All Using Wildcard

An example of doing a query or purge all using wildcard *, is shown in Example 3.

Example 3 Operation, Query, or Purge All Using Wildcard

1. Navigate to the `MtasSubsDataMgmt` MO.
2. Execute MO action `mtasSubsDataMgmtRunQuery` with `mtasSubsDataMgmtRunQueryPui` parameter value *
- or execute MO action `mtasSubsDataMgmtRunPurge` with `mtasSubsDataMgmtRunPurgePui` parameter value *.

Results

The output file contains the following information:



query, 2009-01-10, 13:34:39.974, *

<i><alias pui></i>	<i><default pui></i>	
tel:+3423456789,	sip:+3423456789@someplace.org,	ongoing call (N)
tel:+46123456789,	sip:+46123456789@someplace.com,	ongoing call (N)
sip:+name1@someotherplace.se,	sip:+46123456789@someplace.com,	ongoing call (N)
tel:+46987654321,	sip:+46987654321@someplace2.com,	ongoing call (N)
sip:name2@someplace.eu,	sip:+46987654321@someplace2.com,	ongoing call (N)
tel:+44123456543,	sip:+44123456543@someplace3.com,	ongoing call (N)
sip:name3@someplace.us,	sip:+44123456543@someplace3.com,	ongoing call (N)
tel:+32234567891,	sip:+32234567891@someplace4.com,	ongoing call (N)
tel:+21345234561,	sip:+21345234561@someplace5.co.uk,	ongoing call (N)
processing completed,	9	

purge, 2009-01-10, 13:34:39.974, *

<i><default pui></i>	<i><alias pui></i>
sip:+3423456789@someplace.org,	tel:+3423456789
sip:+46123456789@someplace.com,	tel:+46123456789
sip:+46123456789@someplace.com,	sip:+name1@someotherplace.se
sip:+46987654321@someplace2.com,	tel:+46987654321
sip:+46987654321@someplace2.com,	sip:name2@someplace.eu
sip:+44123456543@someplace3.com,	tel:+44123456543
sip:+44123456543@someplace3.com,	sip:name3@someplace.us
sip:+32234567891@someplace4.com,	tel:+32234567891



sip:+21345234561@someplace5.co.uk, tel:+21345234561
 sip:+2434523456@somewhere.net
 sip:+44gandolf@someotherplace.se
 sip:+pippin@someplace.eu
 sip:samwisegamgee@someplace.us
 sip:777legalos777@someplace.org
 sip:4souron456@somewhere.net
 sip:243452Gandolf@somewhere.net
 processing completed, 13

8.2.4 Query or Purge sip:* or tel:* Using Wildcard

An example of doing a query or purge using wildcard sip:* or tel:*, is shown in Example 4.

Example 4 Operation, Query, or Purge “sip:” or “tel:”*

1. Navigate to the MtasSubsDataMgmt MO.
2. Execute MO action `mtasSubsDataMgmtRunQuery` with `mtasSubsDataMgmtRunQueryPui` parameter value `sip:*` or `tel:*`
 or execute MO action `mtasSubsDataMgmtRunPurge` with `mtasSubsDataMgmtRunPurgePui` parameter value `sip:*` or `tel:*`.

Results for a sip:* Query or Purge

The output file contains the following information:

query, 2009-01-10, 13:34:39.974, sip:*	
<alias pui>	<default pui>
tel:+3423456789,	sip:+3423456789@someplace.org, ongoing call (N)
tel:+46123456789,	sip:+46123456789@someplace.com, ongoing call (N)



sip:+name1@someother place.se,	sip:+46123456789@so meplace.com,	ongoing call (N)
tel:+46987654321,	sip:+46987654321@so meplace2.com,	ongoing call (N)
sip:name2@someplace .eu,	sip:+46987654321@so meplace2.com,	ongoing call (N)
tel:+44123456543,	sip:+44123456543@so meplace3.com,	ongoing call (N)
tel:+32234567891,	sip:+32234567891@so meplace4.com,	ongoing call (N)
tel:+21345234561,	sip:+21345234561@so meplace5.co.uk,	ongoing call (N)
processing completed,	8	

purge, 2009-01-10, 13:34:39.974, sip :*

<default_pui>

<alias_pui>

sip:+3423456789@some
place.org, tel:+3423456789

sip:+46123456789@som
eplace.com, tel:+46123456789

sip:+46123456789@som
eplace.com, sip:+name1@someothe
rplace.se

sip:+46987654321@som
eplace2.com, tel:+46987654321

sip:+46987654321@som
eplace2.com, sip:name2@someplace.
eu

sip:+44123456543@som
eplace3.com, tel:+44123456543

sip:name3@someplace.
us

sip:+32234567891@som
eplace4.com, tel:+32234567891

sip:+21345234561@som
eplace5.co.uk, tel:+21345234561

sip:+2434523456@some
where.net

sip:+44gandolf@someot
herplace.se

sip:+pippin@someplace.
eu



sip:samwisegamgee@so
meplace.us

sip:777legalos777@som
eplace.org

sip:4souron456@somew
here.net

sip:243452Gandolf@so
mewhere.net

processing completed, 14

Results for a tel:* Query or Purge

The output file contains the following information:

query, 2009-01-10, 13:34:39.974, tel:*

<i><alias pui></i>	<i><default pui></i>	
tel:+3423456789,	sip:+3423456789@som eplace.org,	ongoing call (N)
tel:+46123456789,	sip:+46123456789@so meplace.com,	ongoing call (N)
sip:+name1@someother place.se,	sip:+46123456789@so meplace.com,	ongoing call (N)
tel:+46987654321,	sip:+46987654321@so meplace2.com,	ongoing call (N)
sip:name2@someplace .eu,	sip:+46987654321@so meplace2.com,	ongoing call (N)
tel:+44123456543,	sip:+44123456543@so meplace3.com,	ongoing call (N)
sip:name3@someplace. us,	sip:+44123456543@so meplace3.com,	ongoing call (N)
tel:+32234567891,	sip:+32234567891@so meplace4.com,	ongoing call (N)
tel:+21345234561,	sip:+21345234561@so meplace5.co.uk,	ongoing call (N)
processing completed,	9	

purge, 2009-01-10, 13:34:39.974, tel :*

<i><default pui></i>	<i><alias pui></i>
sip:+3423456789@some place.org,	tel:+3423456789



```

sip:+46123456789@som tel:+46123456789
eplace.com,
sip:+46123456789@som sip:+name1 @someothe
eplace.com, rplace.se
sip:+46987654321 @som tel:+46987654321
eplace2.com,
sip:+46987654321 @som sip:name2 @someplace.
eplace2.com, eu
sip:+44123456543@som tel:+44123456543
eplace3.com,
sip:+44123456543@som sip:name3 @someplace.
eplace3.com, us
sip:+32234567891 @som tel:+32234567891
eplace4.com,
sip:+21345234561 @som tel:+21345234561
eplace5.co.uk,
processing completed, 6

```

8.2.5 Query or Purge Using Two Wildcards

An example of doing a query or purge using two wildcards is shown in Example 5.

Example 5 Operation, Query, or Purge Using Two Wildcards

1. Navigate to the MtasSubsDataMgmt MO.
2. Execute MO action `mtasSubsDataMgmtRunQuery` with `mtasSubsDataMgmtRunQueryPui` parameter value `sip:*xxx*` or `tel:*xxx*`

or execute MO action `mtasSubsDataMgmtRunPurge` with `mtasSubsDataMgmtRunPurgePui` parameter value `sip:*xxx*` or `tel:*xxx*`.

Results for a sip:*xxx* or tel:*xxx* Query or Purge

The output file contains the following information:

```

query, 2009-01-10, 13:34:39.974, sip:*xxx* or tel:*xxx*
<alias pui>                <default pui>
tel:+3423456789,           sip:+3423456789@som   ongoing call (N)
                           eplace.org,

```



tel:+46123456789,	sip:+46123456789@so meplace.com,	ongoing call (N)
sip:+name1@someother place.se,	sip:+46123456789@so meplace.com,	ongoing call (N)
tel:+46987654321,	sip:+46987654321@so meplace2.com,	ongoing call (N)
sip:name2@someplace .eu,	sip:+46987654321@so meplace2.com,	ongoing call (N)
tel:+44123456543,	sip:+44123456543@so meplace3.com,	ongoing call (N)
sip:name3@someplace. us,	sip:+44123456543@so meplace3.com,	ongoing call (N)
tel:+32234567891,	sip:+32234567891@so meplace4.com,	ongoing call (N)
tel:+21345234561,	sip:+21345234561@so meplace5.co.uk,	ongoing call (N)
processing completed,	9	

purge, 2009-01-10, 13:34:39.974, sip:*xxx* or tel:*xxx*

<i><default pui></i>	<i><alias pui></i>
sip:+3423456789@some place.org,	tel:+3423456789
sip:+46123456789@som eplace.com,	tel:+46123456789
sip:+46123456789@som eplace.com,	sip:+name1@someothe rplace.se
sip:+46987654321@som eplace2.com,	tel:+46987654321
sip:+46987654321@som eplace2.com,	sip:name2@someplace. eu
sip:+44123456543@som eplace3.com,	tel:+44123456543
sip:+44123456543@som eplace3.com,	sip:name3@someplace. us
sip:+32234567891@som eplace4.com,	tel:+32234567891
sip:+21345234561@som eplace5.co.uk,	tel:+21345234561
sip:samwisegamgee@so meplace.us	



```

sip:777legalos777@som
eplace.org
processing completed,      8

```

8.2.6 Query or Purge Using Multiple Wildcards

An example of doing a query or purge using Query/Purge using multiple wildcards is shown in Example 6.

Example 6 Operation, Query, or Purge with Multiple Wildcards

1. Navigate to the MtasSubsDataMgmt MO.
2. Execute MO action `mtasSubsDataMgmtRunQuery` with `mtasSubsDataMgmtRunQueryPui` parameter value `sip:*xxx*xxx*` or `tel:*xxx*xxx*`

or execute MO action `mtasSubsDataMgmtRunPurge` with `mtasSubsDataMgmtRunPurgePui` parameter value `sip:*xxx*xxx*` or `tel:*xxx*xxx*`.

Results for sip:*xxx*xxx* or tel:*xxx*xxx*

The output file contains the following information:

```

query, 2009-01-10, 13:34:39.974, sip:*xxx*xxx* or tel:*xxx*xxx*
<alias pui>                <default pui>
processing completed,      0

purge, 2009-01-10, 13:34:39.974, sip:*xxx*xxx* or tel:*xxx*xxx*
<default pui>                <alias pui>
sip:samwisegamgee@so
meplace.us
processing completed,      1

```