

System Administrators Guide for Virtual and Cloud Deployment

Ericsson Dynamic Activation 1

SYSTEM ADMINISTRATION GUIDE

Copyright

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

Disclaimer

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

Trademark List

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



Contents

1	Introduction	1
1.1	Purpose and Scope	1
1.2	Target Groups	1
1.3	Typographic Conventions	1
1.4	Prerequisites	1
2	System Information	3
2.1	System Architecture	3
2.1.1	Dynamic Activation Configurations	7
2.2	Directory Structure	8
2.2.1	Description of /opt	8
2.2.2	Description of /var	9
2.2.3	Description of /home	9
2.2.4	Description of /usr	10
2.2.5	Description of /etc	10
2.3	Log Files	10
2.3.1	Operating System	11
2.3.2	Dynamic Activation Application Logs	12
2.3.3	Ericsson SNMP Agent	16
2.3.4	Processing Log File Format	16
2.4	Ports	25
2.4.1	Traffic Ports	25
2.4.2	Operation and Maintenance Ports	26
2.5	Version Identification	27
2.5.1	Software Version	27
2.6	Users	27
2.6.1	System Users	28
2.6.2	Dynamic Activation Users	29
2.7	Crontab	30
3	User Privileges	33
3.1	Administrator in activation Group	33
3.2	Administrator in wheel Group	34
4	Operations	35
4.1	Operation Commands	35
4.2	Dynamic Activation Property Configuration	37
4.3	Dynamic Activation System Status	40
4.3.1	All Nodes Status for 3PP Processes	41



4.3.2	Status of Dynamic Activation Applications	41
4.3.3	Status of License Agent	43
4.4	VM Management	44
4.4.1	Remove VM from Load Balancer	44
4.4.2	Add VM from Load Balancer	46
4.4.3	VM Power Off	48
4.5	Controlled VM Reboot	48
4.6	Enabling and Disabling Provisioning Traffic and CAI3G Test Traffic	48
4.6.1	Enabling and Disabling Provisioning Traffic	48
4.6.2	Enabling and Disabling CAI3G Test Traffic Ports and Provisioning Traffic	49
4.7	Starting and Stopping	50
4.7.1	Dynamic Activation Application Processes	51
4.7.2	Starting and Stopping 3PP Processes	51
4.7.3	Dynamic Activation Modules	53
4.7.4	Increasing Startup Time-out	54
4.8	System Checks	54
4.8.1	Health Check on Dynamic Activation	55
4.8.2	Network Setup Check	58
4.8.3	Maintaining SSL Certificates	60
5	System Administration	61
5.1	Keepalived Cluster Configuration Administration	61
5.1.1	Keepalived Configuration File	61
5.1.2	Logging	61
5.1.3	HAProxy Configuration File	61
5.2	Set Up Persistent Block Storage on VMs (KVM/VMware)	61
5.3	Set Up Ephemeral Storage on VM Instances (CEE)	62
5.4	Configuring ESA	63
5.4.1	Alarms	63
5.4.2	Fault Management Agent	64
5.4.3	CPU Load Monitoring	67
5.4.4	Configuring Internal Heartbeat Monitoring	71
5.5	IPtables	71
5.6	DiffServ	72
5.7	Configuring SSL	73
5.7.1	General Information	73
5.7.2	Remarks and Deviations	73
5.7.3	Configuration Instruction	74
5.8	Configuring Authentication Rules of GUI Users	83
5.8.1	Brute Force Protection Algorithm	83
5.8.2	View Authentication Parameters	83
5.8.3	Modify an Authentication Parameter	84
5.8.4	Restore an Authentication Parameter	85



5.9	License Keys Administration	86
5.9.1	Requesting New License Key	86
5.9.2	Installing License Key	87
5.9.3	Back up License Keys	88
5.9.4	Restoring a Backup of License Keys	89
5.9.5	Removing a License Key	89
5.10	Cassandra Administration	91
5.10.1	Cassandra Data Consistency	92
5.10.2	Cassandra Performance Monitoring	92
5.11	PC Keys	93
5.11.1	Installing a PC Key	93
5.11.2	Rolling Back a PC Key	94
5.12	Processing Log Admin Tool	95
5.12.1	Using Processing Log Admin Tool	95
5.12.2	Configuring Processing Log Admin Tool	97
5.13	Daily Export of Processing Logs	98
5.14	Log File Maintenance	98
5.14.1	Removal of Old Processing Logs	99
5.14.2	Configuration of the Retained Amount of Processing log days	99
5.14.3	Configuration of Processing Logs	99
5.14.4	Configuration of Flat Processing Log Files	102
5.14.5	Other Log Files	104
5.15	Automatic Queue Cleanup for Resilient Activation	104
5.15.1	How the Cleanup Function Works	104
5.15.2	Configure the Cleanup Function	104
5.16	Enabling and Disabling Hostname in CAI3G Error Details	105
5.17	Configuring CAI3G Max Sessions	106
5.18	Configuring MDV Time-Out	107
5.19	Synchronization Among Clusters	107
5.19.1	Verification of Connections	108
5.19.2	Exchanging SSH Key between Clusters	109
5.19.3	Forward SSH Key to New Cluster	112
5.19.4	Exchanging SSH Key When Replacing a Cluster	113
5.19.5	Update Password for Cluster Synchronization	114
5.19.6	License Counters Synchronization	115
5.19.7	Remove Clusters from Synchronization	115
5.20	Dynamic Activation Application Custom Log4j Settings	117
5.20.1	Deploy Custom Log Settings	117
5.20.2	Undeploy Custom Log Settings	118
5.20.3	List Active Custom Log Settings	119
5.21	Configuring External OpenID Connect Provider	119
5.21.1	Pre-request for External OpenID Connect Provider	120
5.21.2	Setting External OpenID Connect Provider Configuration in Dynamic Activation	121
5.21.3	Removing External OpenID Connect Provider Configuration	123



5.22	Asynchronous Request Handler Settings	124
5.22.1	Configuration of Asynchronous CAI3G 1.2 Interface	124
5.23	Batch Handler Settings	126
6	UDC Specific Information	129
6.1	UDC Specific Operations	129
6.1.1	Installing and Upgrading HSS Validator Plug-in	129
6.1.2	Uninstalling and Rolling Back HSS Validator Plug-in	131
6.2	UDC Specific System Administration	133
6.2.1	Activating CLI	133
6.2.2	Notification Rules File Administration	136
6.2.3	Error Mapping File Administration	137
6.2.4	Configuring HTTP Max Connections	139
6.2.5	Configuration of MML Interface	139
6.2.6	Configuration of MML over SSH Interface	144
6.2.7	Configuration of CAI Interface	146
6.2.8	Configuration of EDIFACT Interface	149
6.2.9	Load Default NE Groups and Routing Methods	152
6.3	UDC Data Durability	152
6.3.1	Manual Replay Operation	152
6.3.2	Operational Instruction in the Event of a Failed Replay Operation	155
7	Resource Configuration Specific Information	159
7.1	Use Housekeeping Script to Remove Obsolete Device Configurations	159
7.1.1	How The Housekeeping Script Works	159
7.1.2	Remove Obsolete Device Configurations	160
8	Maintenance Routines	161
8.1	Daily Maintenance	161
8.2	Weekly Maintenance	161
8.3	Monthly Maintenance	161
	Reference List	163



1 Introduction

This section is an introduction to this document. It contains information about the prerequisites, purpose, scope, and target group for the document. This section also contains explanations of typographic conventions used in this document.

1.1 Purpose and Scope

This document covers the system administration, operation, and maintenance routines available for Ericsson Dynamic Activation (EDA) in a virtualized and cloud deployment.

1.2 Target Groups

The target groups for this document are as follows:

- System Administrator
- Network Administrator
- Network Supervision Administrator

1.3 Typographic Conventions

Typographic conventions are described in the document *Library Overview*, Reference [1].

For information about abbreviations and terms used throughout this document, refer to *Glossary of Terms and Acronyms*, Reference [2].

1.4 Prerequisites

The following are the prerequisites to make full use of this document:

- Knowledge about the Ericsson™ Dynamic Activation (EDA) product
- Knowledge about Linux operating system
- Knowledge about Kernel Based Virtualized Machine (KVM)
- Knowledge about VMware
- Knowledge about Ericsson Cloud Execution Environment (ECEE)



2 System Information

This section holds information about the Dynamic Activation system, and how to retrieve useful hardware and software information.

2.1 System Architecture

This section describes the Dynamic Activation system architecture, as shown in Figure 1.

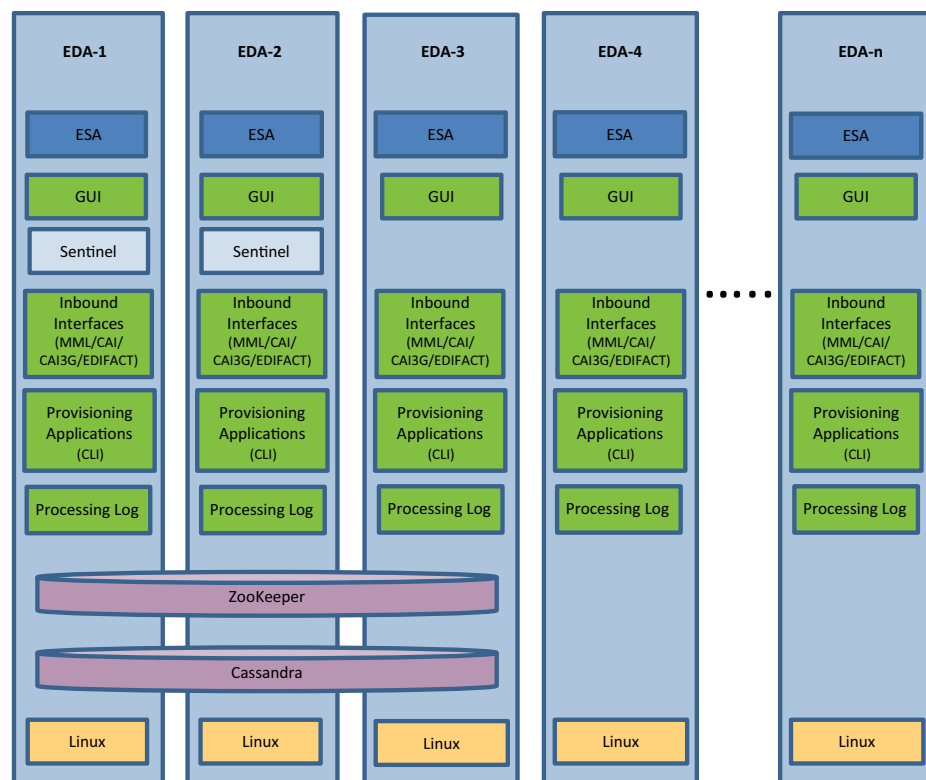


Figure 1 Dynamic Activation System Architecture

The system consists of Virtual Machines (VMs), deployed on one or several physical hosts.

ZooKeeper

ZooKeeper is a cluster storage engine that provides mechanism to get notifications when data is modified, internal data distributor for configuration (for example NEs, NE Groups, Routing), symmetric data modifications, user management, session replication, group membership, and license management.



Zookeeper runs in a separate process and its servers run (by default) on three VMs in the cluster. For redundancy, it is recommended that the zookeeper instances run on separate physical hosts.

For ZooKeeper to work, at least two ZooKeeper instances need to be operational at the same time. Stopping of ZooKeeper instances must be handled with caution, since the Dynamic Activation application is depending on its function. If more than one ZooKeeper instance is brought down simultaneously, the Dynamic Activation application will not function properly.

Restarting one ZooKeeper instance during traffic could cause short traffic disturbance.

Cassandra

Cassandra is a distributed database providing data access through the Cassandra Query Language (CQL) interface. Cassandra is used for storage and presentation of provisioning events performed by the Business Logic, these are the Dynamic Activation processing logs.

Cloud-init

Cloud-init handles early initialization of a cloud instance. It is used to set instance hostname, setting SSH keys, configure network interfaces and adding entries in the `/etc/hosts` file.

Puppet

Puppet is a configuration management system that automatically enforces the correct state of each node. Every Puppet agent contacts the Puppet master within a time interval (default value is 30 seconds), and sends facts about the node's configuration. The Puppet master uses these facts to compile detailed data about how the node should be configured and sends it back to the Puppet agent, which makes the necessary changes to enforce the node's desired state. Finally the Puppet agent sends a report back to the Puppet master with any changes made to the node's configuration.

Keepalived

Keepalived is a routing software which provides simple and robust facilities for load-balancing, and High Availability to Linux systems. It relies on the HAProxy load balancer solution.

HAProxy

HAProxy is a solution that offers High Availability, load balancing, and proxying for TCP and HTTP-based applications.

Function deployment in Dynamic Activation



The Dynamic Activation functions are implemented as Module and 3PP software, used by the modules. For each module, there is a corresponding process (JVM), deployed on one or several nodes, see Table 2.

The functions showed in Figure 1 are implemented by modules, see Table 1.

Table 1 *Functions and Modules in Dynamic Activation*

Function	Module
Inbound Interfaces (MML)	dve-inbound-interfaces-application
Inbound Interfaces (CAI/CAI3G/EDIFACT)	nbia-module
GUI	tomcat-server
Provisioning Applications	dve-application
	arh-module
	akka-cluster-service-lib
	activation-orchestration-module
	lockservice-module
	ema-pq-module
Processing Log Admin Tool	admin-tool
CUDB Block Proxy	cudb-block-proxy
Fault Management service	ema-faultmgmt-module
Authentication service	ema-authentication-module
Batch Handler service	ema-batch-module

- The `dve-inbound-interfaces-application` module supports the inbound MML Interface, mediating incoming requests to different services.
- The `nbia-module` supports the CAI3G, the CAI, and the EDIFACT inbound interfaces, mediating incoming requests to different services.
- The `tomcat-server` module hosts the GUI function. It is based on the Apache Tomcat 3PP.
- The `dve-application` module executes provisioning logic, converting service invocations into provisioning requests towards the NEs, for example HLR, CUDB, HSS. It also supports massive updates, and multi-destination provisioning activities through the CLI.
- The `arh-module` provides support for asynchronous requests. When an asynchronous service request is received, the `arh-module` module generates a notification to the service consumer. Once the service request is processed by the business logic, the module generates a provisioning result notification.
- The `akka-cluster-service-lib` module implements the internal Service Location Register (SLR) function. Service provider modules register their services in the SLR when the module is started. Service consumer modules request service Lookups from the SLR, when needed.



- The `activation-orchestration-module` provides support for stateful and resilient execution of model driven activation logic and queueing towards provisioning resource endpoints.
- The `lockservice-module` provides support for preventing the concurrent provisioning to the same subscriber. When more than one request is received on the same subscriber, the first request acquires a lock and other requests are rejected with an error code.
- The `ema-pq-module` provides support for asynchronous queues to process failed commands. Commands for a subscriber in the same NE are executed in sequence.
- The `admin-tool` provides an import for storing and an export function of the stored processing logs generated by the different modules.
- The `cudb-block-proxy` mediates incoming blocking and unblocking requests into the `dve-application` module from the CUDB. Upon a blocking request, the `dve-application` module holds certain provisioning requests towards the CUDB, until a deblocking request is received, on which the provisioning requests are resumed.

For information about which provisioning requests that are blocked, see *Configuration Manual for Resource Activation*, Reference [8].

The backup notification from CUDB is received through the O&M VLAN in the O&M IP address. It is also necessary to configure the JMX ports in the right order, `<VIP-OAM-IP>:8994:8099`

- The `ema-faultmgmt-module` is responsible of receiving faults from all other modules. The faults are propagated from this module to ESA for sending SNMP alarms, and also sent out as SMTP if email notifications are configured.
- The `ema-authentication-module` is responsible for authentication of users. Other modules in the system that need to authenticate user data, interacts with this module.
- The `ema-batch-module` provides support for bulk provisioning of CAI and synchronous CAI3G requests towards the NEs, it interacts with `nbia-module` to execute provisioning requests.

The deployment of modules and 3PPs is shown in Table 2 and Table 3.

Table 2 Module Deployment in Dynamic Activation

Module	Node
<code>nbia-module</code>	All nodes
<code>dve-inbound-interfaces-application</code>	All nodes
<code>akka-cluster-service-lib</code>	All nodes
<code>cudb-block-proxy</code>	node-1 only



Module	Node
tomcat-server	All nodes
dve-application	All nodes
arh-module	All nodes
admin-tool	All nodes
ema-faultmgmt-module	All nodes
ema-authentication-module	All nodes
activation-orchestration-module	All nodes
lockservice-module	All nodes
ema-pq-module	All nodes
ema-batch-module	All nodes

Table 3 3PP Deployment in Dynamic Activation

3PP	Node
Linux/Red Hat Enterprise Linux 7	All nodes
Apache Tomcat	All nodes
ZooKeeper	node-1, node-2, and node-3
Cassandra	node-1, node-2, and node-3
License Server	node-1 and node-2
Ericsson SNMP Agent (ESA)	All nodes
Cloud-init	All nodes
Puppet	All nodes
Keepalived	node-1 and node-2
HAProxy	node-1 and node-2

2.1.1 Dynamic Activation Configurations

This section describes the Dynamic Activation configurations available for a virtualized and cloud deployment.

The minimum Dynamic Activation system with High Availability consists of three VMs deployed on different hosts.

If more provisioning capacity is needed, additional VMs can be added to the cluster.

For more information on the Dynamic Activation configurations, refer to *Function Specification Resource Activation*, Reference [3].

2.2 Directory Structure

The most important parts of Dynamic Activation directory structure are shown in Figure 2.

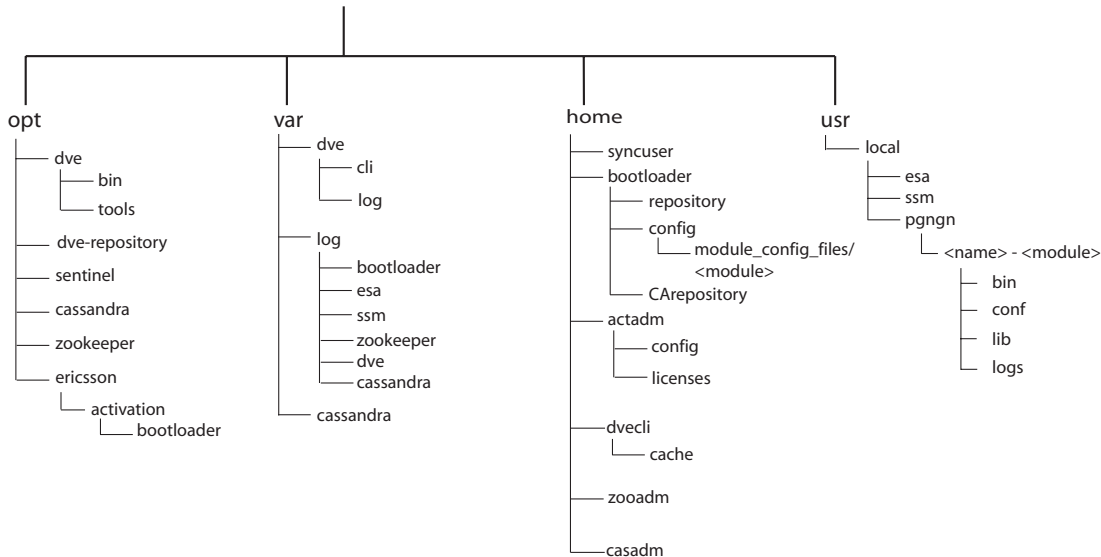


Figure 2 Dynamic Activation Directory Structure

2.2.1 Description of /opt

The folder `/opt` contains add on application software packages. See Table 4 for more specific information on the contents.

2.2.1.1 Description of /opt

The folder `/opt` contains add on application software packages. See Table 4 for more specific information on the contents.

Table 4 Directory: /opt

Directory	Comments
/opt/dve/bin	Dynamic Activation Application scripts
/opt/dve/tools	Contains System tools, for example, JMX batch client, multi-log-consolidation
/opt/dve-repository	RPM Software installation files
/opt/sentinel	Sentinel™ installation directory
/opt/cassandra	Cassandra installation directory and configuration
/opt/zookeeper	ZooKeeper server and configuration
/opt/ericsson/activation/bootloader	Bootloader static files



2.2.2 Description of /var

The folder `/var` contains variable files, such as logs and spool files. See Table 5 for more specific information.

Table 5 Directory: `/var`

Directory	Contents
<code>/var/dve/cli</code>	CLI result files
<code>/var/log</code>	System log files
<code>/var/log/bootloader</code>	Bootloader log files
<code>/var/log/esa</code>	ESA log files
<code>/var/log/ssm</code>	SSM log files
<code>/var/log/zookeeper</code>	ZooKeeper log files, commit log and database
<code>/var/log/cassandra</code>	System log for Cassandra
<code>/var/log/dve</code>	Dynamic Activation system- and, custom log files
<code>/var/cassandra</code>	Cassandra data directory

2.2.3 Description of /home

The folder `/home` contains Dynamic Activation configuration files and shared deployment directories. See Table 6 for more specific information.

Table 6 Directory: `/home`

Directory	Contents
<code>/home/dveinstaller</code>	Dynamic Activation application configuration files and installation logs
<code>/home/bootloader</code>	Bootloader home folder
<code>/home/bootloader/repository</code>	Software repository of Dynamic Activation modules and submodules
<code>/home/bootloader/config</code>	Automatically generated internal Dynamic Activation configuration files
<code>/home/actadm/config</code>	Persisted Dynamic Activation configuration files
<code>/home/actadm/config/log</code>	Log for configuration file changes
<code>/home/actadm/licenses</code>	Licenses (node-1 or node-2)
<code>/home/bootloader/config/module_config_files/<Module></code>	Module configuration files with parameters that, individually, cannot be supported by the <code>bootloader.py</code> CLI
<code>/home/bootloader/CArepository</code>	All used customer adaptations
<code>/home/dvecli/cache</code>	Used by Dynamic Activation Scheduled Procedures to cache temporary results



Directory	Contents
/home/casadm	Home directory for Cassandra administrator user
/home/zooadm	Home directory for Zookeeper administrator user

2.2.4 Description of /usr

The folder `/usr` application software packages. See Table 7 for more specific information on the contents.

Table 7 Directory: /usr

Directory	Contents
/usr/local/esa	ESA installation and configuration
/usr/local/ssm	SSM installation and configuration
/usr/local/pgngn	Base directory for all Dynamic Activation modules, see Table 2. Dynamic-log-consolidation logs

2.2.5 Description of /etc

The folder `/etc` application software packages. See Table 8 for more specific information on the contents.

Table 8 Directory: /etc

Directory	Contents
/etc/cloud	Cloud-init configuration files
/etc/puppet	Puppet configuration files and modules
/etc/keepalived	Keepalived configuration file
/etc/haproxy	Configuration file for the HAProxy load balancer

2.3 Log Files

This section contains information about different log files in the system.



Note: Personal data consists of information related to an identifiable person. An identifiable person is one that can be identified indirectly, for example by its MSISDN, IMSI, or passport number, or directly, for example name, date of birth and postal address. Dynamic Activation handles personal data. Sometimes that personal data can be saved in certain log and result files. Those files are automatically maintained, to make sure that personal data is not retained too long. For more information, see Section 5.14.5 on page 104.

Sensitive information such as passwords, is not recorded in log files.

- Information about how the log files are rotated in the `nbia-module` is available in the following configuration files on each processing node (node-3 to node-n).
 - `/usr/local/pgngn/<module>-<version>/etc/org.ops4j.pax.logging.cfg`: The configuration of how module log files (including debug information, exceptions, and errors) are rotated.
 - `/usr/local/pgngn/<module>-<version>/config/proclog-log4j.properties`: The configuration of Processing Log.
- Information about how the log files are rotated in all other modules is available in each corresponding `log4j.xml` file. These XML files are located in the `/usr/local/pgngn/<module>-<version>/config` directory, on each processing node.

As a default setting, the logs have a maximum size of 15 MB before they are rotated, and stores a maximum of 10 log files before the oldest logs are deleted.

2.3.1 Operating System

The following log files are available:

- `/var/log/messages`

This is the Linux log file. It is rotated when larger than 4096 KB.

- `/var/log/auth`

This is the log file for system logon attempts.

Note: For a full list of messages, check all nodes.



2.3.2 Dynamic Activation Application Logs

Log File Sizes

All the log files introduced in this section have a maximum size of 15 MB before they are rotated, and a maximum of 10 log files are stored before the oldest logs are deleted.

When a log file gets rotated, it is backed up with the same name, but appended with a “1”, “2”, “3”, and then stepping on.

For example, `massiveoperations.log` is updated to `massiveoperations.log.1`, and then `massiveoperations.log.2`.

10 rotated files can exist at the same time, meaning when the 11th log file gets rotated, it overwrites the previous log file appended with a “10”.

Module Log Files

Modules in Dynamic Activation have their respective log files, which are located in `/var/log/dve`.

The log files use the following name convention:

`<module-name>[-<log-type>].log`, where:

- `<module-name>` indicates which module the log file belongs to.
- (Optional) `<log-type>` can be omitted, or one of the following, depending on what log information is contained:
 - Omitted: contains common log information.
 - `audit`: contains failed login attempts to the Dynamic Activation.
 - `config`: contains configuration changes in the GUI, personal data, or both.
 - `console`: contains console output of this module.
 - `karaf`: contains special log for NBIA.
 - `oauth`: contains GUI login attempts and management of GUI users.

For more information, see Table 9.

Log file name-examples: `arh-module.log`, `arh-module-console.log`.

Table 9 Applicable Log Types of Modules

Module	.log	audit	config	console	karaf	oauth
dve-inbound-interfaces-application	Yes	Yes	No	Yes	No	No



Table 9 *Applicable Log Types of Modules*

Module	.log	audit	config	console	karaf	oauth
nbia-module	Yes	No	No	Yes	Yes	No
tomcat-server	Yes	Yes	Yes	No	No	Yes
dve-application	Yes	Yes	Yes	Yes	No	No
arh-module	Yes	No	No	Yes	No	No
akka-cluster-service-lib	Yes	No	No	Yes	No	No
activation-orchestration-module	Yes	No	No	Yes	No	No
admin-tool	Yes	No	No	No	No	No
cudb-block-proxy	Yes	No	No	Yes	No	No
ema-faultmgmt-module	Yes	No	No	Yes	No	No
ema-authentication-module	Yes	Yes	No	Yes	No	No
lockservice-module	Yes	No	No	No	No	No
ema-pq-module	Yes	No	No	Yes	No	No
ema-batch-module	Yes	No	No	Yes	No	No

Special Log Files on All Nodes

The following log files are available on all nodes:

- `/var/log/dve/consolidated.log`

This file logs exceptions and errors in the application, and can also be used for debugging the application by changing the log levels in the corresponding `log4j.xml`, or `org.ops4j.pax.logging.cfg` file.

The `nbia-module` is the module that uses the `org.ops4j.pax.logging.cfg` config file. All other modules use the `log4j.xml` logging config file.

These files can contain personal data.

- `/var/log/dve/multi-log-consolidation-console.log`

This log file contains information of the multi-log-consolidation process. Under normal operating conditions, this file is empty.

- `/var/log/dve/massiveoperations.log`

Note: This file is only applicable for UDC solutions.

This log file contains all messages sent over CLI.

This file can contain personal data.

- `/var/log/dve/requests/failedrequests.log`



This log file contains failed requests.

- `/var/log/dve/request/partiallysucceededoperations.log`

This log file contains the partially successful request and the subsequent southbound operations.

If the sequence of operations was not successful, this causes inconsistency in the CUDB. Correct the inconsistency manually. For more information, refer to *CUDB Subscription Repair and Remove Procedures*, Reference [4].

This file can contain personal data.

- `/var/log/dve/request/eir_partiallysucceededoperations.log`

Note: This file is only applicable for UDC solutions.

This log file contains the partially successful CAI3G request and the subsequent southbound operations.

These southbound operations relate to the logging of IMEIlogdata in the CUDB. The unsuccessful operation causes incomplete data in the CUDB and requires to be manually corrected through direct southbound requests towards the CUDB. For information on how to perform such manual procedures, refer to the **EIR** section in *CUDB Subscription Repair and Remove Procedures*, Reference [4].

This file can contain personal data.

- `/var/log/dve/request/successfulrequests.log`

This log file contains successful requests.

This file can contain personal data.

2.3.2.1 Log Level Settings

It is possible to change the log level on the following log files:

- `<module_name>-audit.log`
- `failedrequests.log`
- `partiallysucceededoperations.log`
- `eir_partiallysucceededoperations.log`
- `successfulrequests.log`

The following log level parameters can be set:

- `@LOG_LEVEL_AUTHORIZATION@` - authorization (default set to OFF)



- @LOG_LEVEL_AUTHENTICATION@ - authentication (default set to TRACE)
- @LOG_LEVEL_FAILED_REQUESTS@ - failed requests (default set to OFF)
- @LOG_LEVEL_SUCCESSFUL_REQUESTS@ - successful requests (default set to OFF)
- @LOG_LEVEL_PARTIALLY_SUCCEEDED_REQUESTS@ - partially succeeded requests (default set to INFO)
- @LOG_LEVEL_PARTIALLY_SUCCEEDED_EIR_OPERATIONS@ - partially succeeded EIR operations (default set to INFO)

The following log level values are supported and can be modified by editing the corresponding config file.

Table 10 Log Level Values in Dynamic Activation

Level	Description
TRACE	Logs all messages. This mode should not be used in production.
DEBUG	Logs debug messages. This mode should not be used when PGNGN is in production.
INFO	Logs informational messages, similar to "verbose" mode.
WARN	Logs potentially harmful events and situations, where the application is carried on as usual.
ERROR	Logs error events where the application can still function.
FATAL	Logs severe error events where the application might quit abnormally.
OFF	Logs nothing.

Follow the procedure to configure the log levels:

1. Run the following command to configure log level setting.

From node-1:

```
$ bootloader.py config set --parameter <log_level_parameter> --value <log_level_value>
```

2. From node-1, run the following command, for all nodes in the cluster, one by one, to activate the change:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

2.3.3 Ericsson SNMP Agent

The following log files are used in ESA:

- `/var/log/esa/MasterAgent.log`

This log file contains log entries related to the Master Agent operations.

- `/var/log/esa/FmAgent.log`

This log file contains log entries related to the FM Agent operations.

- `/var/log/esa/PmAgent.log`

This log file contains log entries related to the PM Agent operations.

- `/var/log/esa/alarms.log`

This log file contains every alarm and event that has been triggered in the FM Agent, which means all `alarm raise`, `alarm clear` and `event` are logged.

- `/var/log/esa/nontranslated/<ipaddress>.log`

This log file is non-translated alarms log and contains every trap and notification that has gone through the alarm translation process in the FM Agent without being successfully matched to a condition. The complete trap content is logged in a log file named `<ipaddress>.log`, where `<ipaddress>` is the IP address of the alarming source.

- `/usr/local/ssm/log`

This directory holds the log file `ssmagent.log`, containing information about ESA System Service Monitor (SSM) operations and errors.

2.3.4 Processing Log File Format

Processing log records both provisioning, and GUI operations in Dynamic Activation.



It is possible to view processing logs in the **Log Management** subtab in Dynamic Activation GUI, see *User Guide for Resource Activation, Reference* [7]. Processing logs are indexed by instance; for example, `imsi`, `msisdn` to easier find related processing logs bound to a certain subscriber.

To extract processing logs from the database, use the Processing Log Admin Tool. For detailed information, see Section 5.12 on page 95. The same file structure is needed when importing processing logs.

Log files are strictly following Comma-Separated Value (CSV) file format. CSV uses a comma to separate values, and uses a double quote (") character around fields that contain reserved characters, such as commas or new lines. If the field itself contains double quote, that double quote should be escaped by one double quote (") character.

Attributes not used for provisioning will not be encoded to a readable format in the processing log.

The structure of log files is described as follows:

For each log file, there is a file header describing what value one log entry contains. The head contains the following field names:

`LogType (v1.1)`, `RootLogId`, `SubLogId`, `TransactionId`, `Instance`, `Operation`, `Status`, `User`, `Hostname`, `Protocol`, `Target`, `StartTime`, `ExecuteTime`, `ResponseCode`, `FullRequest`, `FullResponse`, `ReplayLogId`

Table 11 *Field Name*

Field Name	Description	Log Level
<code>LogType (v1.1)</code>	<p>Describes the type of log a log entry has. Current possible values are:</p> <ul style="list-style-type: none"> <code>northbound</code>, for provisioning inbound processing log. <code>internal</code>, for Business Logic processing log. <code>southbound</code>, for provisioning outbound processing log. 	<p>The following table list depicts which log levels apply for each entry:</p> <ul style="list-style-type: none"> 1-3 3 2 and 3 <p>For information about the different log levels, see Table 23.</p>



Field Name	Description	Log Level
RootLogId	<p>Describes root log identification of one log entry and it is globally unique. It is composed according to this convention <code><hostname><datetime><sequence number><protocol></code>.</p> <ul style="list-style-type: none">• <code>hostname</code>, is the hostname of Dynamic Activation• <code>date</code>, follows this syntax: <code>yyMMdd</code>.• <code>time</code>, follows this syntax: <code>Hhmmss</code>.• <code>sequence number</code>, is generated by Dynamic Activation from 0001 to 9999. If it has reached 9999, 0001 is the next one.• <code>protocol</code>, defines which protocol is used, for example, MML or CAI3G.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
SubLogId	Describes different parts of the system recording processing log. For example “/1”, “/1/2”, “/1/2/1”. One slash stands for one different part, and the number in each part defines how many operations that have been executed.	Not relevant for northbound log type.
TransactionId	<p>Describes transaction identification.</p> <p>The <code>TransactionId</code> is not currently used in this version but is still logged for backwards compatibility towards AS, it is passed by <code>soapOperation</code> to be logged as string.</p>	-
Instance	<p>Describes instance identification and context in the message header of a provisioning request, such as <code>msisdn=12345678:imsi=87654321:context=SH376896987</code></p> <p>The <code>Instance</code> of Batch Handler also contains batch job id, such as <code>batchJobId=b502bfd b-ee7e-49ae-bc9b-dc642f673f9</code></p>	-
Operation	Describes what operation a log entry has, such as CREATE, SET, GET, and DELETE.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
Status	Indicates processing result, possible values are: SUCCESSFUL, FAILED, and CANCELLED.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
User	Shows specific user activities, such as which user did send a certain provisioning request, or which user did perform a specific GUI operation.	Not relevant for southbound log type.
Hostname	Is the hostname of Dynamic Activation.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
Protocol	Defines which protocol is used, such like GUI, MML, and CAI3G1_2, and more.	Not relevant for internal log type.
Target	Defines the object type that is operated on. This could mean the actual MML command in MML or <code>MOType</code> in CAI3G on the northbound, or the NE name southbound.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
StartTime	Records start time of processing.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.



Field Name	Description	Log Level
ExecuteTime	Stands for total execution time of processing.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
ResponseCode	Stands for response code of processing.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3. Response code 0 indicates a successful response.
FullRequest	Describes the detailed request, such as complete MML request or CAI3G request.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
FullResponse	Describes the detailed response, such as complete MML request or CAI3G response.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.
ReplayLogId	Reference to source RootLogId and SubLogId for a replay operation.	Always holds a value, irrespective of log level, that is applicable for log-levels 1–3.

The following is an example of a processing log file:

LogType(v1.1),RootLogId,SubLogId,TransactionId,Instance,Operation,Status,User,Hostname,Protocol,Target,StartTime,ExecuteTime,ResponseCode,FullRequest,FullResponse,ReplayLogId

```
"southbound","vn-11507091326370003CAI3G1_2","/1/1/1/1","","","SEARCH","FAILED","",
"vn-1","LDAP","FT_CUDB","2015-07-09 13.26.37.307","00 00:00:00.001","32",
"dn: serv=Identities,IMSI=264000000510112,dc=imsi,ou=identities,dc=operator,dc=com
Search Scope: OBJECT
Search Filter: (objectClass=*)
Time Limit Milliseconds: 0
Count Limit: 0
Return Object: False
DerefAliases: ALWAYS
DerefLink: False","[LDAP: error code 32 - No Such Object]","",
```

```
"southbound","vn-11507091326370003CAI3G1_2","/1/1/1/1","","","SEARCH","FAILED","",
"vn-1","LDAP","FT_CUDB","2015-07-09 13.26.37.309","00 00:00:00.001","32",
"dn: IMSI=264000000510112,dc=imsi,ou=identities,dc=operator,dc=com
Search Scope: OBJECT
Search Filter: (objectClass=*)
Time Limit Milliseconds: 0
Count Limit: 0
Return Object: False
DerefAliases: NEVER
DerefLink: False","[LDAP: error code 32 - No Such Object]","",
```

```
"southbound","vn-11507091326370003CAI3G1_2","/1/1/1/1/1","","","SEARCH","FAILED","",
"vn-1","LDAP","FT_CUDB","2015-07-09 13.26.37.311","00 00:00:00.001","32",
"dn: serv=Identities,MSISDN=4919018510112,dc=msisdn,ou=identities,dc=operator,dc=com
Search Scope: OBJECT
Search Filter: (objectClass=*)
Time Limit Milliseconds: 0
Count Limit: 0
Return Object: False
DerefAliases: ALWAYS
DerefLink: False","[LDAP: error code 32 - No Such Object]","",
```

```
"southbound","vn-11507091326370003CAI3G1_2","/1/1/1/1/1/1","","","SEARCH","FAILED","",
"vn-1","LDAP","FT_CUDB","2015-07-09 13.26.37.313","00 00:00:00.001","32",
"dn: serv=CSPS,MSISDN=4919018510112,dc=msisdn,ou=identities,dc=operator,dc=com
Search Scope: OBJECT
Search Filter: (objectClass=*)
Time Limit Milliseconds: 0
Count Limit: 0
```



3/1543-CSH 109 628 Uen D | 2017-09-14

3/1543-CSH 109 628 Uen D | 2017-09-14



3/1543-CSH 109 628 Uen D | 2017-09-14

23

```
SOCFNRY: 0
SOCFU: 0
SODCF: 0
SOCLIP: 0
SOCLIR: 0
SOCOLP: 0","[LDAP: error code 68 - Entry Already Exists]",
"vn-11507091326370003CAI3G1_2/1/1/1/1/1/1/1/1/1/1/1/1/1"

"southbound","vn-11507091326380001REPLAY","/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1","","","ADD","FAILED","",
"vn-1","LDAP","FT_CUDB_REPLAY","2015-07-09 13.26.38.328","00 00:00:00.001","68",
"dn: ei=gprs,serv=CSPS,mscId=4622814443244bbba03ffb88f586f4f3,ou=multiSCs,dc=operator,dc=com
changetype: add
objectClass: top
objectClass: alias
objectClass: CUDBExtensibleObject
aliasedObjectName: PDPCP=0,ou=gprsProfiles,serv=CSPS,ou=mscCommonData,dc=operator,
dc=com","[LDAP: error code 68 - Entry Already Exists]",
"vn-11507091326370003CAI3G1_2/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1"

"southbound","vn-11507091326380001REPLAY","/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1","","","ADD","FAILED","",
"vn-1","LDAP","FT_CUDB_REPLAY","2015-07-09 13.26.38.330","00 00:00:00.001","68",
"dn: ei=camel,serv=CSPS,mscId=4622814443244bbba03ffb88f586f4f3,ou=multiSCs,dc=operator,dc=com
changetype: add
objectClass: top
objectClass: alias
objectClass: CUDBExtensibleObject
aliasedObjectName: CSP=0,ou=camelProfiles,serv=CSPS,ou=mscCommonData,dc=operator,dc=com",
"[LDAP: error code 68 - Entry Already Exists]",
"vn-11507091326370003CAI3G1_2/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1/1"

"northbound","vn-11507091326380001REPLAY","","","timestamp=1436441194","AUTOREPLAY","SUCCESSFUL",
"replayer","vn-1","","","2015-07-09 13.26.38.076","00 00:00:00.299","0",
"REPLAY:STARTTIME=1436441194,ENDTIME=1436441198;","Automatic Replay operation successful",""
```

Example 1 Example of Processing log file

2.3.4.1 Cassandra

Processing logs from all nodes are, upon provisioning, saved in the Cassandra cluster. The logs are stored as an aggregated view, to be viewable in the **Log Management** subtab in Dynamic Activation GUI. Each processing log is replicated into, by default, three copies that are saved on different nodes. The spread algorithm for where processing logs are replicated too, is handled by Cassandra, and by that not controlled or possible to know from a Dynamic Activation perspective.

As Processing logs are stored, Cassandra determines and discovers how to efficient compress data, by comparing it. This means that Cassandra will compress both new Processing logs and old Processing logs in a more efficient way, after some usage.

2.3.4.2 Flat Processing Log File

The flat processing log files are named as `flatlog-moduleName.log_<hostName>`, for example: `flatlog-nbia.log_node`, and are stored in the default directory `/var/log/dve/flatlog/` on each node. Each module is corresponding to a specialized Log Type, see Table 12.

When the flat processing log file reaches the maximum file size or after a time interval, the file gets rotated and it is backed up to a CSV formatted file with the name `<flatlogfileName> <data> <time> <sequenceNumber>.csv`, for



example: flatlog-nbia.log_node1_20160407_015025_00001.csv.
This CSV file exists until the end of the life cycle.

The setting of the flat processing log files can be modified in bootloader, see Section 5.14.4 on page 102 for all the configurable parameters and their default values.

Table 12 Module Name and Log Type for Flat Processing Log File

Module Name	Log Type	Log File
nbia-module	Northbound(CAI/CAI3G) and Notification (for asynchronous CAI3G request)	<FLATLOG_PATH_NBIA>/flatlog-nbia.log_<HostName>
dve-inbound-interfaces-application	NorthBound (MML)	<FLATLOG_PATH_DVE_INBOUND_INTERFACES_APPLICATION>/flatlog-dve-inbound-interfaces-application.log_<HostName>
dve-application	Southbound and Internal	<FLATLOG_PATH_DVE_APPLICATION>/flatlog-dve-application.log_<HostName>
tomcat-server	Northbound (GUI operation)	<FLATLOG_PATH_TOMCAT_SERVER>/flatlog-tomcat-server.log_<HostName>
arh-module	Internal (Asynchronous CAI3G)	<FLATLOG_PATH_ARH>/flatlog-arh.log_<HostName>

2.4 Ports

This section includes information about the different traffic, and O&M ports used in Dynamic Activation.

2.4.1 Traffic Ports

The following table contains the traffic ports.

Table 13 Dynamic Activation Traffic Ports

Port	Protocol	Interface	Description	Module
8010	Telnet	eth1/eth2	MML traffic	dve-inbound-interfaces
8111	SSH	eth1/eth2	MML traffic	dve-inbound-interfaces
3010	EDIFACT	eth1/eth2	EDIFACT traffic	nbia-module
3300	Telnet	eth1/eth2	CAI traffic	nbia-module
3301	Telnet	eth1/eth2	CAI traffic with SSL	nbia-module
8080	HTTP/TCP	eth1/eth2	CAI3G traffic and Async CAI3G traffic	nbia-module



Port	Protocol	Interface	Description	Module
8181	HTTPS/TCP	eth1/eth2 (1)	CAI3G traffic and Async CAI3G traffic	nbia-module
8888	HTTP/TCP	* (2)	CAI3G test traffic and Async CAI3G test traffic	nbia-module
8989	HTTPS/TCP	* (2)	CAI3G test traffic and Async CAI3G test traffic	nbia-module

(1) If using separate interfaces for provisioning traffic and O&M traffic respectively, eth2 is used. If the same network interface is used for both, eth1 is used.

(2) Means all interfaces.

2.4.2 Operation and Maintenance Ports

The Operation and Maintenance ports are described in Table 14:

Table 14 Dynamic Activation Operation and Maintenance Ports

Port	Protocol	Interface	Description	Module
22	SSH	* (1)	Used for secure access and data transfer	-
123	NTP	*(1)	Used for time synchronization	-
161, 162	SNMP	*(1)	Used by ESA for handling O&M over SNMP	-
2828	Telnet	eth0	SLR CLI	akka-cluster-service-lib
8009	RMI	*(1)	JMX interface number 2 used for internal monitoring of application deployment and status.	-
8023	Telnet (2)	*(1)	Massive CLI	dve-application
8099	RMI	*(1)	JMX interface number 1 for runtime configuration of Blocking CUDB Before Backup only.	cudb-block-proxy
8100	RMI	*(1)	JMX interface number 1 for runtime configuration.	dve-application
8140	HTTPS	*(1)	Puppet	-
8282 (3)	HTTP	eth1	O&M GUI access	tomcat-server
8383	HTTPS	eth1	O&M GUI access, secure	tomcat-server
8904	RMI	*(1)	JMX interface number 1 used for internal monitoring of application deployment and status.	-
8992	Telnet	*(1)	Massive CLI with SSL	dve-application



Port	Protocol	Interface	Description	Module
8994	RMI	*(1)	JMX interface number 2 for runtime configuration of blocking CUDB before back up only.	cudb-block-proxy
8995	RMI	*(1)	JMX interface number 2 for runtime configuration	dve-application
9000	HTTP	eth1	HAProxy stats GUI	-
9042	CQL	eth0	Used by applications to obtain stored data from the Cassandra Query interface. Also used by Cassandra's CQLSH client to communicate with the Cassandra database.	All modules creating or modifying processing logs.
9162	HTTP/TC P	*(1)	ESA SSM	-

(1) Means all interfaces.

(2) Configuration instructions are described in Section 6.2.1 on page 133.

(3) The HTTP traffic is automatically redirected to HTTPS on port 8383 by tomcat.

2.5 Version Identification

This section describes how to retrieve information about the different component versions and also how to check the hardware present in the system.

Information about Customer Adaptations on Dynamic Activation is presented with the Dynamic Activation component versions.

2.5.1 Software Version

Print all the installed Dynamic Activation RPMs.

From any node, locally:

```
$ sudo 3ppmon version
```

Print all installed components and Customer Adaptation System Properties for Dynamic Activation.

From node-1:

```
$ bootloader.py system version
```

2.6 Users

This section gives a brief description of the users of Dynamic Activation.



2.6.1 System Users

The following are the default system users for Dynamic Activation:

- `actadm` - belongs to the group `activation` and is used for administering Dynamic Activation.
- `dvecli` - belongs to the group `activation` and is used for fetching massive result-files.
- `casadm` - belongs to the group `activation` and is used for administering the Cassandra database.
- `zooadm` - belongs to the group `activation` and owns the ZooKeeper application.
- `syncuser` - belongs to the group `activation` and is used for synchronization of license counters and configurations by SFTP.

The system users can be administered with standard Linux commands.

System Handling

The system user `actadm` is the user that owns most of the processes and files used by Dynamic Activation. For management of the system, the administrative users `administrators` need to be created, see section **Create Administrative User**, Section 2.6.1.1 on page 29. Administrators are members of the `activation` group, and are allowed to perform operation and maintenance of the system.

Locking Mechanism

Failed login attempts for all system users are logged in `/var/log/secure` on each server.

If too many sequential login attempts fail, the user is locked and an event is raised. For more information about alarms and events, refer to *Event and Alarm Handling*, Reference [5].

To view the failed login attempts status of a user, enter the following command as user `root`:

```
# pam_tally2
```

If a user is locked, it can be unlocked by entering the following command as user `root`:

```
# pam_tally2 --user <user_name> --reset
```

The number of unsuccessful login attempts required before an event is raised can be edited in the file `/etc/puppet/modules/pam_tally2/files/password-auth-ac`



Change the value of the parameter `deny=<deny_value>`

Note: This must be done on the puppet master node. Check `/etc/hosts` on any node to find the puppet master.

2.6.1.1 Create Administrative User

This section contains information on how to create non-root users for administering purposes, such as log file reading, process monitoring, managing Dynamic Activation processes, installation of modules, and more.

To create an administrative user, log in as user `root` on nodes-1 and follow the instructions in the step-list:

1. To create a user, run the following command:

```
# addActivationUser <administrator_username>
```

Note: If the user is to exist on all the nodes, that is to be global, the above command needs to be executed on every VM in the cluster.

2. Set the password and activate the new user:

```
# passwd <administrator_username>
```

2.6.1.2 Delete Administrative User

To delete an administrative user, log in as user `root` on node-1 and follow the instructions in the step-list:

1.

```
# userdel -r <administrator_username>
```

2.6.2 Dynamic Activation Users

Dynamic Activation users can be administered through the GUI. For the details, refer to *User Guide for Resource Activation*, Reference [7].

Failed login attempts for Dynamic Activation users are logged in different files, depending on which interface the login attempt was made for, see Section 2.6.2.1 on page 29.

2.6.2.1 Failed Login Attempts

- Failed login attempts through the MML, CAI, and CAI3G interfaces, are logged in the `/var/log/dve/ema-authentication-module-audit.log` file, residing on the node-1.

Example output:



```
2017-03-28 11:20:12.923 [acs-service-44] INFO AuditLog - [CAI] Lo
has failed
2017-03-28 11:23:13.270 [acs-service-45] INFO AuditLog - [MML] Lo
2017-03-28 11:23:44.797 [acs-service-45] INFO AuditLog - [CAI3G1_
[cai3gtest] has failed
```

- Failed login attempts through the CLI interface are logged in the `/var/log/dve/dve-application-audit.log` file, residing on the nodes.

Example output:

```
2017-03-30 08:17:00,957 [AUTHENTICATIONLOG] User [clitest] failed
2017-03-30 08:17:00,958 [AUTHENTICATIONLOG] loginAttemptsThreshold
```

- Failed login attempts through the GUI interface are logged in the `/var/log/dve/tomcat-server-audit.log` file, residing on the node-1.

Example output:

```
2017-03-30 08:24:43,779 [AUTHENTICATIONLOG] User [testuser] failed
2017-03-30 08:24:54,767 [AUTHENTICATIONLOG] User [testuser] is loc
```

The log files above contain, except for information about failed login attempts, also information about user locking when maximal number of consecutive unsuccessful login attempts is reached.

An event is sent from ESA:

- For every failed login attempt.
- When a user is locked (because of threshold reached).

2.7 Crontab

Dynamic Activation uses the following crontab jobs for housekeeping purposes:

Note: All crontab jobs are applicable for all nodes.

- `auto_erase.sh` - handles certain log and response files.

For more information, see Section 5.14.5 on page 104.

- `proclog-retain-maximum-days.sh` - this script is used to retain the maximum number of days configured. It removes the N+1:th day configured to retain a steady number of preferred days with logs in the system. If there



are more logs than the N+1:th days, they won't be removed until the next run of the script. This script is run hourly and increments per node, and is only executed by the `actadm` user.

For more information, see Section 5.14.2 on page 99.

To list all crontab jobs, run the following command as user `root`:

```
# crontab -l -u actadm
```





3 User Privileges

Certain system administration commands require `superuser` privileges to run. It is highly recommended to setup system administrator users part of the `wheel` group, and run such commands with `sudo` privileges instead of using the `root` account.

As an example, the `pam_tally2` command :

- (High security risk) If logged in as user `root`:
`[root@node-1 ~]# pam_tally2`
- (Low security risk/Better choice) If logged in as a system administrator:
`[<System Admin>@node-1 ~]$ sudo pam_tally2`

For security reasons, do not use `root` user for Operations and Management (O&M) of Dynamic Activation. Instead, create Dynamic Activation administrators, that are part of the `activation` group, to perform O&M operations. See Section 3.1 on page 33.

It is highly recommended to disable the `root` user, and replacing this account with one or more System Administrator users that are part of the `wheel` group. See Section 3.2 on page 33.

3.1 Administrator in activation Group

The activation group defines trusted system administrators that are allowed to manage Dynamic Activation.

System administrators in the `activation` group have privileges to perform the following O&M:

- Use all available `bootloader` commands, for example, manage application modules, set `bootloader` configuration, and so on.
- Manage 3PPs with `3ppmon` commands by using `sudo`.
- Run scripts, for example, `syncuser/sync_ssh_keys`, `replay-admin-tool`, `housekeeping`, and so on.

Unlike users in the `wheel` group, the `activation` users cannot use `sudo` to execute other commands that require `superuser` privileges, except the `3ppmon` commands.

For instruction on how to create `activation` users, see Section 2.6.1.1 on page 29.



3.2 Administrator in wheel Group

System administrators in the `wheel` group can use `sudo` to execute all commands that require `superuser` user privileges. They have privileges to perform:

- Initial deployment and installation
- User management of Linux users, for example, create new users, set passwords, and so on.
- Manage License Keys in Sentinel.
- License Agent administration
- Reboot cluster
- Health check (due to accessing system variables)
- Operating System level management

To keep certain operation-procedures consistent in the Dynamic Activation documentation, some commands (that require `superuser` privileges to run), are specified to be run as user `root` with `#` prompt. These commands can instead, and preferably, be executed as a `wheel` group user by adding `sudo` prior to the commands.

4 Operations

This section describes system operation commands, such as starting and stopping the system, and how to perform various system checks.

Note: All the commands should be used depending on the state of the system. If the system is running traffic, some of the following commands can disturb the provisioning.

4.1 Operation Commands

This section describes the following commands which are used for checking system status and performing system operations.

- `3ppmon` commands – Manage 3rd party applications and services that the Dynamic Activation Application processes depend on.
- `bootloader.py` commands – Manage the Dynamic Activation Application progresses (also known as modules).
- `systemctl` commands – Manage system background processes (also known as daemons).

Note: `3ppmon` commands must be run with `superuser` privileges.

Table 15 3ppmon Options

Command	Description
<code>3ppmon startesa stopesa</code>	Used to start or stop the ESA processes.
<code>3ppmon startssm stopssm</code>	Used to start or stop the SSM process.
<code>3ppmon startzookeeper stopzookeeper</code>	Used to start or stop the ZooKeeper service. ZooKeeper runs by default on node-1, node-2, and node-3.
<code>3ppmon startcassandra stopcassandra</code>	Used to start or stop Cassandra.
<code>3ppmon startlserv stoplserv</code>	Used to start or stop the Sentinel License Server.
<code>3ppmon status status --host all status -H all status --host <hostname> status -H <hostname></code>	Used to view the status for all nodes and the detailed status for one specified node (including port status, deploy status, process status and alarm status for ESA, SSM, and Sentinel). It is possible to replace hostname with IP address in the command.
<code>3ppmon version</code>	Used to view installed version of platform and other installed RPM packages on the system.

Note: `bootloader.py` commands are exclusively run on node-1.

**Table 16** *Bootloader Supported Commands*

Command	Description
<code>bootloader.py system version</code>	Used to list system product and module version information, for each node in the cluster.
<code>bootloader.py node activate --host <ip/hostname>[HOST1 HOST2 HOST-n]/all> ⁽¹⁾</code>	Used to, again, activate installed modules and submodules applicable for the given host.
<code>bootloader.py node status --host <ip/hostname/all> --detailed</code>	Used to list status info for modules and submodules.
<code>bootloader.py node start/stop --host <ip/hostname/all></code>	Used to administer Dynamic Activation applications, except for ZooKeeper, ESA, and Sentinel.
<code>bootloader.py module start s top restart --name/-n <name> --host <ip/hostname/all></code>	Used to start, stop, or restart the process corresponding to the given module.
<code>bootloader.py module add --name/-n <name> --enabled/-e <true/false> --host <ip/hostname></code>	Used to install a specific module. ⁽²⁾
<code>bootloader.py module delete --name/-n <name> --host <ip/hostname></code>	Used to uninstall a specific module. ⁽²⁾
<code>bootloader.py module update --name/-n <module tar.gz file> --host <ip/hostname></code>	Used to update a specific module. If it does not exist it will be added. ⁽²⁾
<code>bootloader.py module getversion --name/-n <module tar.gz file> --host <ip/hostname></code>	Used to get the version of a specific module.
<code>bootloader.py submodule add --name/-n <name> --type/-t <types allowed in parent module> --parent/-p <name> --host <ip/hostname></code>	Used to install a specific submodule. ⁽²⁾
<code>bootloader.py submodule delete --name/-n <name> --parent/-p <name> --host <ip/hostname></code>	Used to uninstall a specific submodule. ⁽²⁾
<code>bootloader.py submodule update --name/-n <submodule tar.gz file> --type/-t <types allowed in parent module> --parent/-p <name> --host <ip/hostname></code>	Used to update a specific submodule. If it does not exist it will be added. ⁽²⁾
<code>bootloader.py submodule deploy --name/-n <submodule tar.gz file> --type/-t <types allowed in parent module> --parent/-p <name> --host <ip/hostname/all></code>	Used to deploy a specific submodule (only supported for type jdv).
<code>bootloader.py submodule undepl oy --name/-n <submodule tar.gz file> --parent/-p <name> --host <ip/hostname/all></code>	Used to undeploy a specific submodule (only supported for type jdv).
<code>bootloader.py config list [--show_all/-A]</code>	Used to list configuration information for the system.
<code>bootloader.py config set --parameter/-p <parameter> --value/-v <value></code>	Used to override the value of a parameter with a new customized value. ⁽²⁾
<code>bootloader.py config remove --parameter/-p <parameter></code>	Used to restore the default value of a parameter.



Command	Description
<code>bootloader.py logging deploy -log --name/-n <(sub)module name> --jarfile/-j <jarfile> --level/-l <loglevel> --host <ip/hostname/all> [--outputfile/-o <log file name>]</code>	Used to deploy custom log settings for Dynamic Activation Application sub-modules.
<code>bootloader.py logging undepl y-log --name/-n <(sub)module name> --jarfile/-j <jarfile> --host <ip/hostname/all></code>	Used to undeploy custom log settings for Dynamic Activation Application sub-modules. ⁽³⁾
<code>bootloader.py logging undeploy-log --name/-n all --host all</code>	Used to quickly undeploy all custom log settings for all JAR-files/modules on all hosts.
<code>bootloader.py logging deploy-proclog --name/-n <module name> --category/-c <proclog category></code>	Used to deploy custom proclog logging settings on a module. ⁽⁴⁾
<code>bootloader.py logging undeploy-proclog --name/-n <(sub)module name> --category/ -c <proclog category></code>	Used to undeploy custom proclog logging settings on a module. ⁽⁴⁾
<code>bootloader.py logging list [--type/-t <proclog/log/all>] [--name/-n <(sub)module name>]</code>	Used to list all active custom logs/proclogs settings on the Dynamic Activation system.

(1) *all* is only used for new installations. It should not be used in any other cases.

(2) For these commands to take effect, they need to be activated. A `bootloader.py node activate --host <ip/hostname/all>` command is required.

(3) This command will undeploy per jar-file and on one or several hosts depending on parameter input.

(4) This is used in Customer Adaptation (CA) development.

Note: `systemctl` command is run locally on every node, and must be run with superuser privileges.

Table 17 Systemctl Commands

Command	Description
<code>systemctl start stop status license_agent</code>	Used to start or stop the License Agent process, or check its status.

4.2 Dynamic Activation Property Configuration

Dynamic Activation properties are categorized as follows:

- Site-specific properties must be adapted to specific sites (such as IP address), opposed to non-site specific.
- Adaptable properties implies that the customer can change the value at operation time, opposed to non-adaptable.

This implies the following:



- The configuration of non-adaptable site-specific properties is part of the Dynamic Activation installation procedure, handled by the installation scripts.
- Configuration of adaptable properties is performed at operation time, by use of the `bootloader.py` command.

To configure a property with a customized value, run the following commands from node-1:

1. To list the adaptable properties:

```
$ bootloader.py config list -A
```

2. Configure the selected adaptable property:

```
$ bootloader.py config set --parameter @<Adaptable  
Property>@ --value <value>
```

3. Activate the modification for each node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

To restore a property to its default (initially installed) value, run the following commands from node-1:

1. To list the adaptable properties:

```
$ bootloader.py config list -A
```

2. Restore the selected adaptable property:

```
$ bootloader.py config remove --parameter @<Adaptable  
Property>@
```

3. Activate the modification for each node, one by one:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: Repeat the command for each node affected by this change.

To check which properties that have been customized, run the following command, from node-1:

```
$ bootloader.py config list
```



```
Configuration attributes for nbia-module:
  No user configuration exist

Configuration attributes for admin-tool:
  No user configuration exist

Configuration attributes for akka-cluster-service-lib:
  No user configuration exist

Configuration attributes for tomcat-server:
  No user configuration exist

Configuration attributes for ema-faultmgmt-module:
  No user configuration exist

Configuration attributes for NCA-Application:
  No user configuration exist

Configuration attributes for ema-authentication-module:
  No user configuration exist

Configuration attributes for dve-application:
  No user configuration exist

Configuration attributes for lockservice-module:
  No user configuration exist

Configuration attributes for activation-orchestration-module:
  No user configuration exist

Configuration attributes for dve-inbound-interfaces-application:
  No user configuration exist

Configuration attributes for jdv-cudb-config-assembly:
  No user configuration exist

Configuration attributes for cudb-block-proxy:
  No user configuration exist

Configuration attributes for arh-module:
  No user configuration exist

Configuration attributes for NotificationRules:
  No user configuration exist

Configuration attributes for UDC-Application:
  No user configuration exist

Configuration attributes for ema-batch-module:
  No user configuration exist
```

Example 2 Config List Printout

In the example, the `cudb-block-proxy` module has a customized parameter, `@BLOCK_PROXY_SSLCERT_STORE_PASSWORD@`

All other modules have default values for their properties.

4.3 Dynamic Activation System Status

The `3ppmon status` command is used to view the status for the 3PP processes in all nodes, or the detailed status for one specified node.

The `bootloader.py node status` command is used to view the status of Dynamic Activation application processes for all nodes, or the status for one specified node.

4.3.1 All Nodes Status for 3PP Processes

Execute the following command on any node to check the deployment status for all 3PPs in all nodes:

```
$ sudo 3ppmon status --host all
```

Host	ESAPrcs	LicSrv	LVS	ZooKpr	Puppet	Cassandra	ActvAlrm
node-1	UP	UP	UP	UP	UP	UP	0
node-2	UP	UP	UP	UP	UP	UP	0
node-3	UP	-	-	UP	UP	UP	0
node-4	UP	-	-	-	UP	-	0

Example 3 Output for a Dynamic Activation system

Table 18 3PP Processes Attributes

Attribute	Description
Host	The hostname of the node in the cluster.
ESAPrcs	ESA processes. UP means that all ESA processes are running, DOWN means that one or more ESA processes are not running.
LicSrv	Sentinel License server. UP means that the Sentinel License Server is running and DOWN means that the server is not running. A dash (-) means that there is no license server on that node. The Sentinel License server is only running on node-1 or node-2.
LVS	Keepalived process that handles the Virtual IP failover between node-1 and node-2.
ZooKpr	ZooKeeper process. UP means that ZooKeeper is running on that node and DOWN means that it is down. A dash (-) means that there is no ZooKeeper server on that node. The ZooKeeper is running on node-1, node-2, and node-3 by default.
Puppet	Puppet process is used for automating installation and configuration of the virtual Dynamic Activation. Puppet master on node-1(primary) and node-2(backup) sends out configurations to all nodes.
Cassandra	Cassandra process. Up means that the process is running and DOWN means that the process is down. One Cassandra process is running on each of node-1, node-2, and node-3. Cassandra process. Up means that the process is running and DOWN means that the process is down. One Cassandra process is running on each of node-1, node-2, and node-3.
ActvAlrm	Number of active alarms for the node. If there are active alarms, run the <code>factivealarms</code> command on node-1 or node-2, for detailed information about the alarms. Alarms for all nodes are listed when running this command. For more information about the specific alarms, refer to <i>Event and Alarm Handling</i> , Reference [5].

Note: A question mark (?) indicates unknown status.

4.3.2 Status of Dynamic Activation Applications

From node-1, execute the following command to check the status of the Dynamic Activation applications:

```
$ bootloader.py node status --host all
```



Host	Module	Version	Status	Bindings
=====	=====	=====	=====	=====
host-1	akka-cluster-service-lib	3.3.1	Running	OK (3/3)
host-1	activation-orchestration-module	2.2	Running	
host-1	ema-faultmgmt-module	1.15	Running	OK (3/3)
host-1	ema-authentication-module	1.2.3	Running	
host-1	cudb-block-proxy	2.0.4	Running	
host-1	lockservice-module	1.7.12	Running	
host-1	admin-tool	3.138		
host-1	tomcat-server	162.43	Running (18/0)	
host-1	-> DVE-OM-GUI	162.43		
host-1	-> Log-Backend-Rest	162.43		
host-1	-> Log-Consolidation-Web	162.43		
host-1	-> LEH-Backend	162.43		
host-1	-> LooseErrorHandlingWeb	162.43		
host-1	-> BootloaderRest	162.43		
host-1	-> Queue-Management	162.43		
host-1	-> Launchpad	162.43		
host-1	-> OrderManagement	162.43		
host-1	-> OrderManagement-Backend	162.43		
host-1	-> SyncSingleEntryGuiFrontend	162.43		
host-1	-> SyncSingleEntryGuiBackend	162.43		
host-1	-> replayer	162.43		
host-1	-> inbound-interfaces-rest	162.43		
host-1	-> Scm-Web	162.43		
host-1	-> batchhandler-backend	2.4		
host-1	-> batchhandler-frontend	2.4		
host-1	nbia-module	3.138	Running	OK (3/3)
host-1	-> mpe-business-nbi-notification	3.138		
host-1	dve-application	3.138	Running (71/0)	OK (3/3)
host-1	-> NCA-Application	162.43		
host-1	-> jdv-nca-assembly	162.43		
host-1	-> UDC-Application	162.43		
host-1	-> jdv-udc-assembly	162.43		
host-1	-> jdv-cudb-config-assembly	162.43		
host-1	-> NotificationRules	162.43		
host-1	arh-module	162.9	Running	OK (3/3)
host-1	dve-inbound-interfaces-application	162.43	Running	
host-1	ema-batch-module	2.4	Running	
host-2	akka-cluster-service-lib	3.3.1	Running	OK (3/3)
host-2	activation-orchestration-module	2.2	Running	
host-2	ema-faultmgmt-module	1.15	Running	OK (3/3)
host-2	ema-authentication-module	1.2.3	Running	
host-2	lockservice-module	1.7.12	Running	
host-2	admin-tool	3.138		
host-2	tomcat-server	162.43	Running (18/0)	
host-2	-> DVE-OM-GUI	162.43		
host-2	-> Log-Backend-Rest	162.43		
host-2	-> Log-Consolidation-Web	162.43		
host-2	-> LEH-Backend	162.43		
host-2	-> LooseErrorHandlingWeb	162.43		
host-2	-> BootloaderRest	162.43		
host-2	-> Queue-Management	162.43		
host-2	-> Launchpad	162.43		
host-2	-> OrderManagement	162.43		
host-2	-> OrderManagement-Backend	162.43		
host-2	-> SyncSingleEntryGuiFrontend	162.43		
host-2	-> SyncSingleEntryGuiBackend	162.43		
host-2	-> replayer	162.43		
host-2	-> inbound-interfaces-rest	162.43		
host-2	-> Scm-Web	162.43		
host-2	-> batchhandler-backend	2.4		
host-2	-> batchhandler-frontend	2.4		
host-2	nbia-module	3.138	Running	OK (3/3)
host-2	-> mpe-business-nbi-notification	3.138		
host-2	dve-application	3.138	Running (71/0)	OK (3/3)
host-2	-> NCA-Application	162.43		
host-2	-> jdv-nca-assembly	162.43		
host-2	-> UDC-Application	162.43		
host-2	-> jdv-udc-assembly	162.43		
host-2	-> jdv-cudb-config-assembly	162.43		
host-2	-> NotificationRules	162.43		
host-2	arh-module	162.9	Running	OK (3/3)
host-2	dve-inbound-interfaces-application	162.43	Running	
host-2	ema-batch-module	2.4	Running	


```

host-3 akka-cluster-service-lib 3.3.1 Running OK (3/3)
host-3 activation-orchestration-module 2.2 Running
host-3 ema-faultmgmt-module 1.15 Running OK (3/3)
host-3 ema-authentication-module 1.2.3 Running
host-3 lockservice-module 1.7.12 Running
host-3 admin-tool 3.138
host-3 tomcat-server 162.43 Running (16/0)
host-3 -> DVE-OM-GUI 162.43
host-3 -> Log-Backend-Rest 162.43
host-3 -> Log-Consolidation-Web 162.43
host-3 -> LEH-Backend 162.43
host-3 -> LooseErrorHandlingWeb 162.43
host-3 -> BootloaderRest 162.43
host-3 -> Queue-Management 162.43
host-3 -> Launchpad 162.43
host-3 -> OrderManagement 162.43
host-3 -> OrderManagement-Backend 162.43
host-3 -> SyncSingleEntryGuiFrontend 162.43
host-3 -> SyncSingleEntryGuiBackend 162.43
host-3 -> replayer 162.43
host-3 -> inbound-interfaces-rest 162.43
host-3 -> Scm-Web 162.43
host-3 -> batchhandler-backend 2.4
host-3 -> batchhandler-frontend 2.4
host-3 nbia-module 3.138 Running OK (3/3)
host-3 -> mpe-business-nbi-notification 3.138
host-3 dve-application 3.138 Running (71/0) OK (3/3)
host-3 -> NCA-Application 162.43
host-3 -> jdv-nca-assembly 162.43
host-3 -> UDC-Application 162.43
host-3 -> jdv-udc-assembly 162.43
host-3 -> jdv-cudb-config-assembly 162.43
host-3 -> NotificationRules 162.43
host-3 arh-module 162.9 Running OK (3/3)
host-3 dve-inbound-interfaces-application 162.43 Running
host-3 ema-batch-module 2.4 Running

```

Example 4 Printout for all nodes

Table 19 Printout Parameters

Host	The host it concerns.
Module	The Dynamic Activation module (can contain sub-modules).
Version	The module version.
Status	If the module is a process, a process status is shown, Running Stopped. If the process contains sub-modules, deployment status of it's artefacts is shown in a parenthesis. The first number indicates fully deployed artefacts. The second number indicates artefacts that have failed deployment. For example, Running (46/0) indicates that 0 artefacts out of 46 have failed to deploy.
Bindings	This column shows whether the module has services and are registered successfully in the SLRs, OK (Bounded to all SLRs) WARN (Bound in at least one SLR) NOK (Not bounded to any SLR).

4.3.3 Status of License Agent

From node-1, execute the following command to check the status of the License Agent:

```
$ sudo -u actadm lastat
```



Note: It can take up to one hour for the License Agent process to reload new or updated license keys. Forced reload of license keys can be achieved by restarting the License Agent process. See the command listed in Table 17.

Node	ZooKeeper	Sentinel	Instances	License	Cores	License	Valid to Ru
host-1	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES
host-2	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES
host-3	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES
host-4	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES
host-5	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES
host-6	CONNECTED	CONNECTED	INSTALLED		INSTALLED		YES

Example 5 Printout for all nodes

Node	ZooKeeper	Sentinel	Instances	License	Cores	License	Valid to Ru
host-1	DISCONNECTED	DISCONNECTED	NOT INSTALLED		NOT INSTALLED		NO
host-2	DISCONNECTED	DISCONNECTED	NOT INSTALLED		NOT INSTALLED		NO
host-3	DISCONNECTED	DISCONNECTED	NOT INSTALLED		NOT INSTALLED		NO
host-4	DISCONNECTED	DISCONNECTED	NOT INSTALLED		NOT INSTALLED		NO
host-5	-	-	-		-		-
host-6	DISCONNECTED	DISCONNECTED	NOT INSTALLED		NOT INSTALLED		NO

Required license key [FAT1023848/2] is missing. Retry after installing the required key.
Required license key [FAT1023848/3] is missing. Retry after installing the required key.

lastat: Error occurred while reading status from one or more nodes. Make sure License Agent ser

Example 6 Licenses are missing and License Agent is not running on node-5

4.4 VM Management

To perform a controlled shutdown or startup of a VM, follow the subchapters in this section.

4.4.1 Remove VM from Load Balancer

This section contains information on how to make sure that there is no traffic loss when shutting down, rebooting or stopping a VM.

Follow the steps:

1. Use a web browser and log in to the HAProxy Stats GUI:

`http://<VIP-OAM-IP>:9000/haproxy_stats`

Username: **admin**

Password: **Ericsson1**

Note: Make sure to change the password after first login. To change password do as follows:

- 1 On node-1, enter the following command as user `root`:

```
# openssl passwd -1
```

Enter a new password

Re-enter the new password

- 2 Copy the encrypted password.

Note: Be sure to include all characters, and DO NOT copy any white-spaces

- 3 On node-1, edit the HAProxy config file.

As user `root`:

```
# vi /etc/puppet/modules/haproxy/templates/haproxy_cfg.erb
```

Replace the encrypted password on the following line with the newly generated password:

```
user admin password $1$YExmVIdN$8Nn3YP3qCYPCgMOgr7tyb0
```

- 4 Verify that the new password is working by logging in to the HAProxy GUI.

Note: It can take up to 30 seconds for the new password to be functional.

2. Navigate to the specific traffic protocol, and select the check box for the node that is to be rebooted/shutdown/stoped. For example, running CAI3G traffic on node 3:

guilPv4_vip_oam:8282											
		Queue			Session rate			Sessions			
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total
	Frontend				0	6	-	4	7	2 000	91
<input type="checkbox"/>	node-1	0	0	-	0	6		0	7	-	83
<input type="checkbox"/>	node-2	0	0	-	0	5		4	5	-	8
<input checked="" type="checkbox"/>	node-3	0	0	-	0	0		0	0	-	0
	Backend	0	0		0	6		4	7	200	91

Choose the action to perform on the checked servers :

Set state to MAINT

Apply

Choose the action to perform on the checked servers : Set state to MAINT Apply

3. In the drop-down list, choose Set state to Maint as action, and press **Apply**.



4. Perform Step 1 - Step 3 for all traffic ports where the specific node is operating.
5. Continue with VM `poweroff/reboot/node stop` according to specific sections in this document.

4.4.2 Add VM from Load Balancer

This section contains information on how to make sure that traffic runs again after rebooting or starting a VM.

Follow the steps:

1. Use a web browser and log in to the HAProxy Stats GUI:

`http://<VIP-OAM-IP>:9000/haproxy_stats`

Username: **admin**

Password: **Ericsson1**

Note: Make sure to change the password after first login. To change password do as follows:

- 1 On node-1, enter the following command as user `root`:

```
# openssl passwd -1
```

Enter a new password

Re-enter the new password

- 2 Copy the encrypted password.

Note: Be sure to include all characters, and DO NOT copy any white-spaces

- 3 On node-1, edit the HAProxy config file.

As user `root`:

```
# vi /etc/puppet/modules/haproxy/templates/haproxy_cfg.erb
```

Replace the encrypted password on the following line with the newly generated password:

```
user admin password $1$YExmVIdN$8Nn3YP3qCYPCgMOgr7tyb0
```

- 4 Verify that the new password is working by logging in to the HAProxy GUI.

Note: It can take up to 30 seconds for the new password to be functional.

2. Navigate to the specific traffic protocol, and select the check box for the node that has been rebooted/started. For example, running CAI3G traffic on node 3:

guiPv4_vip_oam:8282												
		Queue			Session rate			Sessions				
		Cur	Max	Limit	Cur	Max	Limit	Cur	Max	Limit	Total	
	Frontend				0	6	-	0	7	2 000	97	
<input type="checkbox"/>	node-1	0	0	-	0	6		0	7	-	89	
<input type="checkbox"/>	node-2	0	0	-	0	5		0	5	-	8	
<input checked="" type="checkbox"/>	node-3	0	0	-	0	0		0	0	-	0	
	Backend	0	0		0	6		0	7	200	97	

Choose the action to perform on the checked servers :

Set state to READY ▾

Apply

Choose the action to perform on the checked servers : Set state to READY ▾ Apply

3. In the drop-down list, choose Set state to READY as action, and press **Apply**.



4. Perform Step 1 - Step 3 for all traffic ports where the specific node is operating.

4.4.3 VM Power Off

To power off the VM, follow the step-list:

1. Follow the instructions in Section 4.4.1 on page 44.
2. Execute the following command on the current node:

```
# poweroff
```

4.5 Controlled VM Reboot

Attention!

Reboot only one VM at a time.

Because rebooting two or more VMs concurrently causes Cassandra errors and disturbs the traffic.

Reboot of the other nodes need to be done after node-1 is up and running, otherwise it will cause failures.

To perform a controlled reboot of a VM, follow the step-list:

1. Follow the instructions in Section 4.4.1 on page 44.
2. Execute the following command, as user `root` on the current node:

```
# reboot
```

3. Follow the instructions in Section 4.4.2 on page 46.

4.6 Enabling and Disabling Provisioning Traffic and CAI3G Test Traffic

This section describes how to enable and disable the provisioning traffic, and CAI3G test traffic.

4.6.1 Enabling and Disabling Provisioning Traffic

It is possible to enable and disable provisioning (CAI3G, MML, CAI) traffic on Dynamic Activation. This, to allow maintenance to be performed on a traffic handling processing node, without interfering with live traffic on the Dynamic Activation cluster.



To disable traffic execute the following commands:

1. From node-1, run the following command to disable traffic in the cluster:

```
$ bootloader.py config set --parameter @REGISTER_SERVICES@ --value false
```

2. From node-1, run the following command to activate the configuration:

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

To enable traffic, execute the following commands:

1. From node-1, run the following command to enable traffic in the cluster:

```
$ bootloader.py config set --parameter @REGISTER_SERVICES@ --value true
```

2. From node-1, run the following command to activate the configuration:

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

4.6.2

Enabling and Disabling CAI3G Test Traffic Ports and Provisioning Traffic

It is possible to enable and disable CAI3G test traffic ports, over HTTP/HTTPS separated through test ports, using different port numbers, see Table 13.

The test traffic is disabled by default. For a test scenario, the processing node to test with CAI3G test traffic, is switched into test traffic mode.

To disable provisioning traffic and enable test traffic ports on a processing node, execute the following commands:

1. Test traffic can only be handled by one node at a time. Before switching a processing node into test traffic mode, run the following command on node-1 to check the status of the Dynamic Activation applications, to make sure that there is no other processing node running in test mode.

```
$ bootloader.py node status --host all
```

2. From node-1, run the following command to disable traffic on the node:

```
$ bootloader.py config set --parameter @REGISTER_SERVICES@ --value false
```

3. Enable CAI3G test traffic:



```
$ bootloader.py config set --parameter @REGISTER_TEST_SERVICES@ --value true
```

4. From node-1, run the following command to activate the configurations:

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

5. Run test traffic on test ports 8888 or 8989 (secure).

To enable provisioning traffic and disable CAI3G test traffic ports on a processing node, execute the following commands:

Note: This is performed once the test is finished and the processing node previously set to test traffic mode, is to resume live traffic.

1. From node-1, run the following commands to disable the CAI3G test traffic mode and enable provisioning traffic:

```
$ bootloader.py config remove --parameter @REGISTER_TEST_SERVICES@
```

```
$ bootloader.py config remove --parameter @REGISTER_SERVICES@
```

2. From node-1, run the following command to activate the configurations:

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

4.7 Starting and Stopping

The commands for starting and stopping processes in the Dynamic Activation system are listed in the following sections.

The different parts of the application-dependent processes are listed in separate subsections.

Dynamic Activation module processes and 3PP processes are supervised by SSM. If any of the processes go down, SSM starts it. The process is added or removed automatically in SSM, when the following commands are used.

Note: Stopping processes by using these commands do not trigger any alarms.



4.7.1 Dynamic Activation Application Processes

Starting the Dynamic Activation Application Processes

1. Perform the following command, from node-1, to start the Dynamic Activation application processes on a complete system:

```
$ bootloader.py node start --host all
```

2. Perform the following command, from node-1, to start the Dynamic Activation application processes on a specific node `<hostname>`:

```
$ bootloader.py node start --host <hostname>
```

3. Follow the instructions in Section 4.4.2 on page 46.
4. Verify the status of all the nodes.

To check the status for all nodes, see Section 4.3 on page 40.

Stopping the Dynamic Activation Application Processes

1. Follow the instructions in Section 4.4.1 on page 44.
2. Perform the following command, from node-1, to stop the Dynamic Activation application processes on a complete system:

```
$ bootloader.py node stop --host all
```

3. Perform the following command, from node-1, to stop the Dynamic Activation application processes on a specific node `<hostname>`:

```
$ bootloader.py node stop --host <hostname>
```

4. Verify the status of all the nodes.

To check the status for all nodes, see Section 4.3 on page 40.

4.7.2 Starting and Stopping 3PP Processes

This section covers how to start and stop 3PP Processes.

The commands in this chapter apply only for the node they are executed on.

4.7.2.1 Sentinel

This section covers how to start and stop Sentinel RMS. Applies only nodes node-1 and node-2.



Starting Sentinel RMS (License Server)

To start the license server on node-1 or node-2, execute the following command:

```
$ sudo 3ppmon startlserv
```

Stopping Sentinel RMS (License Server)

To stop the license server on node-1 or node-2, execute the following command:

```
$ sudo 3ppmon stoplserv
```

4.7.2.2 Ericsson SNMP Agent

This section covers how to start and stop Ericsson SNMP Agent (ESA) and the System Service Monitors (SSM) agent (comes with ESA).

Starting Ericsson SNMP Agent

To start ESA, execute the following command:

```
$ sudo 3ppmon startesa
```

Stopping Ericsson SNMP Agent

To stop ESA, execute the following command:

```
$ sudo 3ppmon stopesa
```

Starting SSM

To start the SSM agent, execute the following command:

```
$ sudo 3ppmon startssm
```

Stopping SSM

To stop the SSM agent, execute the following command:

```
$ sudo 3ppmon stopssm
```

4.7.2.3 ZooKeeper

This section covers how to start and stop the ZooKeeper service.



Note: These operations can only be performed on a ZooKeeper node. For an overview of which nodes that have ZooKeeper installed, see Section 2 on page 3.

Starting ZooKeeper

To start ZooKeeper, execute the following command:

```
$ sudo 3ppmon startzookeeper
```

Stopping ZooKeeper

To stop ZooKeeper, execute the following command:

```
$ sudo 3ppmon stopzookeeper
```

4.7.2.4

Cassandra

This section covers how to start and stop the Cassandra processes.

Caution!

It is allowed to stop only one Cassandra instance at a time. For more information, see Section 5.10.1.1 on page 92.

Starting Cassandra

To start Cassandra, execute the following command:

```
$ sudo 3ppmon startcassandra
```

Stopping Cassandra

To stop Cassandra, execute the following command:

```
$ sudo 3ppmon stopcassandra
```

4.7.3

Dynamic Activation Modules

This section covers how to start and stop Dynamic Activation modules.

For modules possible to start and stop on the different nodes, see Example 4 in Section 4.3.2 on page 41.



Note: The status-output shows both modules and sub-modules. Sub-modules start with ->.

Starting Dynamic Activation Modules

A module can be started on a specific node, or on all nodes. To start one or all Dynamic Activation modules, follow the step-list:

1. From node-1, execute the following command:

```
$ bootloader.py module start -n <module> --host  
<hostname> | all
```

2. Follow the instructions in Section 4.4.2 on page 46.

Stopping Dynamic Activation Modules

A module can be stopped on a specific node, or on all nodes. To stop one or all Dynamic Activation modules, follow the step-list:

1. Follow the instructions in Section 4.4.1 on page 44.
2. From node-1, execute the following command:

```
$ bootloader.py module stop -n <module> --host <hostname>  
| all
```

4.7.4 Increasing Startup Time-out

When starting a system with many network elements, the startup sequence may fail with a time-out.

To increase the time-out value, execute the following commands:

1. From node-1, run the following command to disable traffic in the cluster:

```
$ bootloader.py config set --parameter @timeout@ --value  
<timeout_value>
```

Note: The <timeout_value> must be lower than four minutes.

2. From node-1, run the following command to activate the configuration:

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

4.8 System Checks

This section describes system tools and other utility programs used during problem tracing and for maintaining and verifying the Dynamic Activation system. It also describes how to perform a health check on the system.

4.8.1 Health Check on Dynamic Activation

Health checks are automatically run during installation, update and upgrade, and checks the following:

- Node status of Cassandra.
- Status of Sentinel
- Node status of Zookeeper
- The ESA cluster status.
- Active ESA alarms.
- Dynamic Activation status on all nodes.

Note: The automatic health-check tool does not check the Network Setup , if unclear provisioning issues exist or the `/var/log/messages` file for recent error messages.

It is possible to manually run a system health check on any occasion.

The following health-check options are available:

Command	Description
<code>healthcheck.py run -n <task or group of tasks to be executed></code>	Used to execute a health-check task or a group of tasks and display the result in the console.
<code>healthcheck.py get</code>	Used to get all the possible task or group of tasks, along with their descriptions, that can be executed by the <code>healthcheck.py run</code> command.
<code>healthcheck.py show -r <time> -d/--detail</code>	Used to show previous health-check results that was run in a specific time. If not specified it will show the latest executed run. <code>-d/--detail</code> is optional and is used to show result along with detailed information.
<code>healthcheck.py show -l/--list</code>	Used to show all previous health-check results in the system

The step-list below shows an example of a manually run system health-check:

Note: All health-check commands need to be run as user `root`, and can only be executed on node-1 or node-2.

1. Verify that the system status:

```
# healthcheck.py run -n <task or group of tasks to be executed>
```

Example:

```
# healthcheck.py run -n Activation_all_Virtual
```

**Output Example:**

```
=====
Name: Activation_all_Virtual
Description: Verify virtual system status is OK
Result stored as: 2017-05-08_13-31-02
=====
activation_zookeeper      PASS
activation_application     PASS
activation_esa_cluster    PASS
activation_sentinel       PASS
activation_cassandra      PASS
activation_esa_alarms     PASS
activation_esa            PASS
```

To get a list of all possible *<task or group of tasks to be executed>* options, run the following command:

```
# healthcheck.py get
```

Output:

Name	Description
Activation_all_Native	Verify native system status is OK
Activation_all_Virtual	Verify virtual system status is OK
activation_application	Check that all modules is up and running
activation_cassandra	Check that cassandra is up and running properly
activation_drbd	Check ownership of drbd is correct (only for native)
activation_esa	Check that esa is up and running properly
activation_esa_alarms	Check if esa alarms exist
activation_esa_cluster	Check esa cluster status is correct
activation_lde_cluster	Check lde cluster status (only for native)
activation_network	Check that the network (evip) is up and running
activation_platform	Check that all 3pp's is up and running
activation_sentinel	Check that sentinel is up and running properly
activation_zookeeper	Check that zookeeper is up and running properly

2. Run the following command to get detailed information on the health-check:

```
# healthcheck.py show -d
```

Output Example:



```

=====
Name: Activation_all_Virtual
Description: Verify virtual system status is OK
Result stored as: 2017-03-22_11-22-58
=====
activation_sentinel      PASS
Description : Check that sentinel is up and running properly
Messages : Sentinel is up and running on all nodes
activation_esa_alarms    PASS
Description : Check if esa alarms exist
Messages : Active alarms:
activation_cassandra     PASS
Description : Check that cassandra is up and running properly
Messages : Cassandra is up and running on all nodes
activation_esa_cluster   PASS
Description : Check esa cluster status is correct
Messages : Esa cluster verified ok
activation_esa           PASS
Description : Check that esa is up and running properly
Messages : Esa is up and running on all nodes
activation_zookeeper     PASS
Description : Check that zookeeper is up and running properly
Messages : Zookeeper is up and running on all nodes
activation_application    PASS
Description : Check that all modules is up and running
Messages : All modules is up and running properly on all nodes

```

Note: If any of the statuses are not shown as PASS, check the logs in `/var/log/healthcheck/` directory.

Health-check logs store the executing flow of each task. Check the corresponding log if one or more tasks failed during the health-check.

The `healthcheck.log` file stores general information of the health-check execution flow.

3. Make sure that no unclear provisioning issues exist.

Note: This step is only applicable on a running system.

Check point: Check Dynamic Activation provisioning logs for any unclear provisioning issues, for example partly executed logs, see Section 2.3.2 on page 11.

4. Check the `/var/log/messages` file for recent error messages, all nodes:

Note: This step is only applicable on a running system.

```
# cat /var/log/messages
```



Check point: No error messages should exist.

4.8.2 Network Setup Check

This section contains information on how to check the network setup.

1. Check that node-1 is the active load balancer:

```
# ip addr show
```

Check point: Verify from received output that correct VRRP address is bound to eth0, eth1, and eth2 (if traffic separation is used).

2. Check load balancer status.

To check the status of the load balancer login to the virtual OAM IP (VIP_OAM) from a web browser, and verify all services.

Navigate to:

```
http://<VIP-OAM-IP>:9000/haproxy_stats
```

Enter:

```
user: admin
```

```
password: Ericsson1
```




Note: Make sure to change the password after first login. To change password do as follows:

- 1 On node-1, enter the following command:

```
# openssl passwd -1
```

Enter a new password

Re-enter the new password

- 2 Copy the encrypted password.

Note: Be sure to include all characters, and DO NOT copy any white-spaces

- 3 On node-1, edit the HAProxy config file.

As user root:

```
# vi /etc/puppet/modules/haproxy/templates/haproxy_cfg.erb
```

Replace the encrypted password on the following line with the newly generated password:

```
user admin password $1$YExmVIdN$8Nn3YP3qCYPCgMOgr7tyb0
```

- 4 Verify that the new password is working by logging in to the HAProxy GUI.

Note: It can take up to 30 seconds for the new password to be functional

Click on the **Hide 'DOWN' servers** link to see all open ports that are distributed by Keepalived or HAProxy.

Check point: All fields on node-1 and node-2 should be green, and all traffic interfaces should be green on all nodes.

3. Verify that IP Forwarding is enabled (`net.ipv4.ip_forward` is set to 1):

```
# sysctl -p
```

Printout :

```
net.ipv4.ip_forward = 1
```

Check point: Verify that the printout matches the example printout above.



4. Verify that the network bridges are configured according to *Network Description and Configuration for Virtual and Cloud Deployment*, Reference [6].

4.8.3 Maintaining SSL Certificates

Certificates for SSL verification are always time limited. Check the validity of all installed SSL certificates.

On the first node, node-1, go to `/home/bootloader/ssl` and execute the following command:

```
$ /usr/java/latest/bin/keytool -list -v -keystore  
<key_filename> -storepass <password>
```

If necessary, configure new certificates. Refer to Section 5.7 on page 72, and Section 6.2.1 on page 133 for instructions.



5 System Administration

This section holds information on how to administer different parts of the system.

5.1 Keepalived Cluster Configuration Administration

This section describes how to edit the Keepalived configuration.

5.1.1 Keepalived Configuration File

When the Keepalived cluster is started, it uses the data in the `keepalived.conf` file that is located in the `/etc/keepalived` directory. The Keepalived configuration is managed by Puppet, and changes are performed in the Puppet module.

5.1.2 Logging

The Keepalived component logs events to the system log, see Section 2.3.1 on page 11 for details on how to find the system log output.

5.1.3 HAProxy Configuration File

Keepalived handles the movable IP and HAProxy provides the load balancer functionality.

The configuration file can be found in the `/etc/haproxy` directory. This file is managed by Puppet, and changes are performed in the Puppet module.

5.2 Set Up Persistent Block Storage on VMs (KVM/VMware)

It is mandatory for commercial deployments to attach an external block storage to each of node-1, node-2, and node-3. Each external block storage must have a minimum size of *output of Dimensioning Tool*₃.

Note: The term “external” means VM external. A VM external block storage can still be provided by a local disks if desired.

The disk space available on the image itself is only there for demonstration purposes and leight-weight testing.

In a virtual deployment the image itself is not intended to be used for storing persistent data, but should be designed as disposable as possible.



To set up persistent block storage on VMs perform the step-list as user `root`:

1. Ensure that node-1, node-2, and node-3 all have unpartitioned disk attached.
2. Log in to a VM in the cluster and execute:

```
# storage.py setup -h <node-n>
```

Note: `<node-n>` refers to the node that is currently being set up for persistent storage. For example, the first VM, `node-1`.

3. At this point, a reboot of the node `<node-n>` is required.

On `<node-n>`, run:

```
# reboot
```

4. Wait for `<node-n>` to reboot properly, then execute the following command to verify the services that are running:

```
# 3ppmon status --host <node-n>
```

5. Repeat Step 2 through Step 4 for node-1, node-2, and node-3.

5.3 Set Up Ephemeral Storage on VM Instances (CEE)

Because the image itself is not intended to hold persistent data, it is mandatory to attach an ephemeral disk to each of node-1, node-2, and node-3. Each ephemeral disk must have a minimum size of $\frac{\text{output of Dimensioning Tool}}{3}$.

Ephemeral storage size is user defined and specified when performing a Virtual Dynamic Activation Cluster installation. For detailed information, see section **Install Virtual Dynamic Activation Cluster** in *Software Installation for Virtual and Cloud Deployment*, Reference [13].

```
vdb          252:16    0   200G    0 disk
└─vdb1       252:17    0    9.3G    0 part /var/cassandra/commitlog
└─vdb2       252:18    0 190.7G    0 part /var/cassandra/data
```

Example 7 Ephemeral Storage Size

Attention!

It is only node-1, node-2 and node-3 (hosting the Cassandra database) that will be initialized with additional ephemeral storage.



Note: Ephemeral storage is by definition only available throughout the lifecycle of an instance. Any data will be lost when terminating an instance as well as when rebuilding an instance. For more information about ephemeral storage and the different storage options, refer to the **Cloud Execution Environment (CEE) 16A R5A** library in <http://cpistore.internal.ericsson.com/>.

Resize of ephemeral storage post initial deployment is not supported in this release.

5.4 Configuring ESA

This section describes the configuration in ESA.

5.4.1 Alarms

The Ericsson SNMP Agent (ESA) manages the alarm handling in Dynamic Activation. Information about the different alarms can be found in the document *Event and Alarm Handling*, Reference [5].

5.4.1.1 Trap Destination

The trap destination decides where ESA sends the SNMP alarms. If SNMP v2c is used, the destination is identified with IP address and port number. If SNMP v3 is used, more SNMP v3 configuration is required.

For more information, see Reference [9].

To update the trap destination host, add or modify the `oss_ip` attribute in `/etc/puppet/modules/hostsfile/templates/hosts.erb`

For example, modifying `10.61.247.135` to `10.61.247.136`

This update will trigger the puppet module to reconfigure and restart ESA with updated trap destination.

Verify that the `trapDestCfg.xml` in `/usr/local/esa/conf` is updated accordingly. Using the same example as above, the change will generate the following output:

Example Output



```
<trapDestCfg xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.ericsson.com/esa" xsi:schemaLocation=
"http://www.ericsson.com/esa trapDestCfg.xsd">
  <managerDefinition snmpVersion="v2c" active="yes">
    <ip>10.61.247.136</ip>
    <port>162</port>
    <securityName>v1v2ReadWriteSecName</securityName>
    <securityLevel>noAuthNoPriv</securityLevel>
  </managerDefinition>
  <managerDefinition snmpVersion="v3" active="no">
    <ip>127.0.0.1</ip>
    <port>162</port>
    <securityName>v3AuthSHAPrivDESSecName</securityName>
    <securityLevel>authPriv</securityLevel>
  </managerDefinition>
</trapDestCfg>
```

By inactivating the trap destination configuration, ESA disregards reading the configuration and thus does not send any traps to that destination. This setting is useful since the configuration can be kept as is, while it is possible to stop the traps only by deactivating it with a `no`.

5.4.1.2 SNMP Version

The configuration files are stored in `/usr/local/esa/conf`

The default SNMP version is v2c.

Attention!

The use of SNMPv3 with multiple agents, such as when using cluster mode, in a virtual IP (VIP) configuration is not a recommended setup. Since the OSS will target the VIP, which in turn randomly targets the agents, the OSS will have issues handling the different engineIDs of the different agents. If VIP is used with SNMPv3 the OSS must be able to aggregate and correlate O&M data from multiple agents as well as keep track of multiple engineIDs for a single IP (the VIP).

For more information on how to change SNMP version, see Reference [9].

5.4.2 Fault Management Agent

During runtime, the Fault Management (FM) Agent is ready to accept alarm triggers and hold active alarms and alarm data, which is made available to the OSS using SNMP. The technician working locally with the system could benefit from looking at the data captured and produced by the FM Agent.

A number of CLI commands exist that makes the setup and configuration of the FM Agent easier. Also, they are useful when troubleshooting faults in the system by looking at the FM Agent alarm information.



The following CLI commands are available for the FM Agent:

- `fmactivealarms` – To view active alarms.
- `fmsendmessage` – To send messages for triggering an alarm or event, or clearing an alarm.

5.4.2.1 Command: View Active Alarms

This command is used for viewing the active alarms that have been triggered in the FM Agent.

Note: This command must be run on node-1 or node-2.

5.4.2.1.1 Command Format

The command format:

`fmactivealarms`

The command does not take any arguments.

5.4.2.1.2 Command Output

When running the command on node-1 or node-2, if there are active alarms, the command output is a complete list of all active alarms. The following output example shows three active alarms:

```
# fmactivealarms
```



```
Active alarms:
!-----
Module           : ALARMMOD
Error Code       : 11
Resource Id      : 1.1.1.2.72
Timestamp        : Wed Sep 21 09:50:03 CEST 2011
Model Description : This is fault number 1!
Active Description : The component XXX has an error.
Event Type       : 1
Probable Cause   : 1024
Severity         : minor
Orig Source IP   : 10.1.100.88
!-----!
Module           : ALARMMOD
Error Code       : 22
Resource Id      : 1.1.1.2.9
Timestamp        : Wed Sep 21 09:52:31 CEST 2011
Model Description : This is fault number 2!
Active Description : The component YYY has an error.
Event Type       : 1
Probable Cause   : 1024
Severity         : minor
Orig Source IP   : 10.1.100.88
!-----!
Module           : ALARMMOD
Error Code       : 33
Resource Id      : 1.1.1.2.21
Timestamp        : Wed Sep 21 09:54:44 CEST 2011
Model Description : This is fault number 3!
Active Description : The component ZZZ has an error.
Event Type       : 1
Probable Cause   : 1024
Severity         : minor
Orig Source IP   : 10.1.100.88
!-----!
```

Example 8 List of Active Alarms

If there are no active alarms, the command output is only a heading:

```
# fmactivealarms
Active alarms:
```

5.4.2.2 Command: Send Message

This command is used for triggering alarms and events.

5.4.2.2.1 Command Format

The command format:

```
fmsendmessage <action> <moduleId> <errorCode> <resourceId>
["<activeDescr>"] [<sourceIp>]
```

The command attributes are the following:



action	The type of action to trigger.
-r	Trigger raise alarm message.
-c	Trigger clear alarm message.
-e	Trigger event message.
moduleId	The module identity of the alarm to trigger.
errorCode	The error code of the alarm to trigger.
resourceId	The resource identity of the alarming object.
activeDescr	The text to send in the alarm message.
sourceIp	The IP address to use as the alarm originating source.

For example, to clear an alarm with Dynamic Activation as module, Error code 4001 and resource ID.1.1.1.1.2.2.192.168.0.3, issue the following command:

From originating node (the node that triggered the alarm):

```
# fmsendmessage -c PG 4001 .1.1.1.1.2.2.192.168.0.3
"manually cleared"
```

From any node:

```
# fmsendmessage -c PG 4001 .1.1.1.1.2.2.192.168.0.3
"manually cleared" 192.168.0.3
```

5.4.2.2.2 Command Output

If the specified alarm is successfully triggered, the command does not give any output.

If the specified alarm could not be triggered, the command returns an error message.

5.4.3 CPU Load Monitoring

Ericsson SNMP Agent (ESA) is also used for monitoring the CPU load. The Performance Management Agent (PMA) module in ESA can be used for this purpose.

PMA saves every five minutes a 3GPP XML file containing the CPU load for each core on each node. Saved files older than one day are automatically removed.



The configuration files are stored in the `/usr/local/esa/conf` directory, several of them are triggered by the puppet module.

All 3GPP files are stored in the `/var/log/esa/pm3gppXml` directory. The files can be fetched through SFTP.

For an example of a 3GPP file, see Example 9.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<meas CollecFile xmlns="http://www.3gpp.org/ftp/specs/archive/32_series/32.435$meas Collec">
  <fileHeader fileFormatVersion="3GPP_PM_32.435_XML_v11.0.0">
    <fileSender/>
    <meas Collec beginTime="2014-01-06T19:45:00-05:00"/>
  </fileHeader>
  <measData>
    <managedElement/>
    <measInfo>
      <job jobId="CPULoadCounterNode3"/>
      <granPeriod duration="PT300S" endTime="2014-01-06T19:50:00-05:00"/>
      <repPeriod duration="PT300S"/>
      <measType p="1">CPULoadCounterNode3Core8.max</measType>
      <measType p="2">CPULoadCounterNode3Core5.min</measType>
      <measType p="3">CPULoadCounterNode3Core12.max</measType>
      <measType p="4">CPULoadCounterNode3Core9.actual</measType>
      <measType p="5">CPULoadCounterNode3Core12.actual</measType>
      <measType p="6">CPULoadCounterNode3Core1.actual</measType>
      <measType p="7">CPULoadCounterNode3Core12.min</measType>
      <measType p="8">CPULoadCounterNode3Core7.max</measType>
      <measType p="9">CPULoadCounterNode3Core5.max</measType>
      <measType p="10">CPULoadCounterNode3Core9.max</measType>
      <measType p="11">CPULoadCounterNode3Core9.min</measType>
      <measType p="12">CPULoadCounterNode3Core4.max</measType>
      <measType p="13">CPULoadCounterNode3Core10.max</measType>
      <measType p="14">CPULoadCounterNode3Core6.max</measType>
      <measType p="15">CPULoadCounterNode3Core3.min</measType>
      <measType p="16">CPULoadCounterNode3Core11.min</measType>
      <measType p="17">CPULoadCounterNode3Core6.min</measType>
      <measType p="18">CPULoadCounterNode3Core5.actual</measType>
      <measType p="19">CPULoadCounterNode3Core8.min</measType>
      <measType p="20">CPULoadCounterNode3Core7.actual</measType>
      <measType p="21">CPULoadCounterNode3Core7.min</measType>
      <measType p="22">CPULoadCounterNode3Core11.max</measType>
      <measType p="23">CPULoadCounterNode3Core6.actual</measType>
      <measType p="24">CPULoadCounterNode3Core2.min</measType>
      <measType p="25">CPULoadCounterNode3Core4.actual</measType>
      <measType p="26">CPULoadCounterNode3Core11.actual</measType>
      <measType p="27">CPULoadCounterNode3Core2.max</measType>
      <measType p="28">CPULoadCounterNode3Core4.min</measType>
      <measType p="29">CPULoadCounterNode3Core10.actual</measType>
      <measType p="30">CPULoadCounterNode3Core2.actual</measType>
      <measType p="31">CPULoadCounterNode3Core1.max</measType>
      <measType p="32">CPULoadCounterNode3Core8.actual</measType>
      <measType p="33">CPULoadCounterNode3Core10.min</measType>
      <measType p="34">CPULoadCounterNode3Core3.actual</measType>
      <measType p="35">CPULoadCounterNode3Core1.min</measType>
      <measType p="36">CPULoadCounterNode3Core3.max</measType>
      <measValue measObjLdn="">
        <r p="1">0</r>
        <r p="2">0</r>
        <r p="3">0</r>
        <r p="4">0</r>
        <r p="5">0</r>
        <r p="6">0</r>
        <r p="7">0</r>
        <r p="8">0</r>
        <r p="9">1</r>
        <r p="10">0</r>
        <r p="11">0</r>
        <r p="12">0</r>
        <r p="13">0</r>
        <r p="14">1</r>
        <r p="15">0</r>
        <r p="16">0</r>
      </measValue>
    </measInfo>
  </measData>
</meas CollecFile>
```



```

    <r p="17">0</r>
    <r p="18">0</r>
    <r p="19">0</r>
    <r p="20">0</r>
    <r p="21">0</r>
    <r p="22">0</r>
    <r p="23">0</r>
    <r p="24">0</r>
    <r p="25">0</r>
    <r p="26">0</r>
    <r p="27">0</r>
    <r p="28">0</r>
    <r p="29">0</r>
    <r p="30">0</r>
    <r p="31">0</r>
    <r p="32">0</r>
    <r p="33">0</r>
    <r p="34">0</r>
    <r p="35">0</r>
    <r p="36">0</r>
  </measValue>
</measInfo>
</measData>
<fileFooter>
  <measCollec endTime="2014-01-06T19:50:00-05:00"/>
</fileFooter>
</measCollecFile>

```

Example 9 3GPP XML File

The following is to be performed on all nodes.

Configuration of PMA:

- Edit the `/etc/puppet/modules/esa_client/templates/mainCfg_xml.erb` file with the following command.

As user root:

```
# vi /etc/puppet/modules/esa_client/templates/mainCfg_xml.erb
```

To configure:

- The output directory.

Configure the `pm3gppXml` node to appropriate output directory. For example:

```

<output>
  <pm3gppXml>/var/log/esa/pm3gppXml</pm3gppXml>
</output>

```

- The time interval in days of how long the 3GPP files are saved.

Configure the `fileRetentionPeriod` node. For example:



```
<pm>
...
<fileRetentionPeriod>1</fileRetentionPeriod>
...
</pm>
```

- The time interval in minutes of how often the system scans for old 3GPP files.

Configure the `fileRetentionScan` node. For example:

```
<pm>
...
<fileRetentionScan>30</fileRetentionScan>
...
</pm>
```

- Edit the `/etc/puppet/modules/esa_client/templates/CPUloadCounterNode_xml.erb` file with the following command.

As user `root`:

```
# vi /etc/puppet/modules/esa_client/templates/CPUloadCounterNode_xml.erb
```

To configure:

- The time interval in seconds of how often the system reads the data.

Configure the `interval` attribute in all `dataSource` nodes. For example:

```
<dataSource interval="60">
```

There are limitations on allowed values. They are 10, 15, 20, 30, 60, 120, 300, 600, 900, 1200, 1800, 3600 seconds. For more details, see [ESA SAG Performance Management](#).

- Edit the `/etc/puppet/modules/esa_client/templates/CPUloadJobNode_xml.erb` file with the following command.

As user `root`:

```
# vi /etc/puppet/modules/esa_client/templates/CPUloadJobNode_xml.erb
```

To configure:

- The time interval in minutes of how often the system collects data before any 3GPP file is written to file.

Configure the `<granularityPeriod>` node. For example:

```
<defGranularityPeriod>5</defGranularityPeriod>
```



There are limitations on allowed values. They are 5, 15, 30, 60 minutes. For more details, see ESA SAG Performance Management.

For the new configuration to take effect, restart ESA:

```
$ sudo 3ppmon stopesa
$ sudo 3ppmon startesa
```

5.4.4 Configuring Internal Heartbeat Monitoring

The PMA module in ESA is used for monitoring the internal heartbeat. Two PMAs are running, one on node-1 and one on node-2. They monitor all nodes and raise an alarm if a node is unavailable.

Counters are fetched by the PMA, by calling an external script, `/opt/dve/bin/getNodesNumber.sh`

Default interval is 60 seconds. Intervals allowed are 10, 15, 20, 30, 60, 120, 300, 600, 900, 1200, 1800, 3600 seconds.

Configuration of heartbeat interval:

1. Edit the `/etc/puppet/modules/esa_master/files/pmNodeCounter.xml` file with the following command:

Note: This step is performed as user `root` on both node-1 and node-2.

```
# vi /etc/puppet/modules/esa_master/files/pmNodeCounter.xml
```

Configure the `dataSource interval` attribute. For example:

```
<dataSource interval="120">
```

2. For the new configuration to take effect, restart ESA:

```
$ sudo 3ppmon stopesa
$ sudo 3ppmon startesa
```

If one of the nodes goes down, an event is sent indicating which particular node that is not responding.

Each counter is published through the `ERICSSON-ESA-PM-MIB` and is possible to fetch through the SNMP `Get` operations from OSS-RC.

5.5 IPtables

This section describes the basic IPtables configuration.



The IPtables configuration is managed by Puppet and changes are done in the Puppet module.

To print all active IPV4 rules, run the following command as user `root`:

```
# iptables -L -vn
```

To print all active IPV6 rules, run the following command as user `root`:

```
# ip6tables -L -vn
```

5.6 DiffServ

The DiffServ framework defined by the IETF is used to implement a set of forwarding treatments and communicate the classification of packets between UDC nodes and backbone/site infrastructure.

Login as user `root` on appropriate nodes and perform the following:

1. To activate DiffServ and DSCP, issue the following commands from the appropriate nodes:
 - Default DiffServ Code Point is 16 (CS2):

```
# iptables -t mangle -A OUTPUT -j DSCP --set-dscp 16
```
 - Default DiffServ Code Point for LDAP traffic towards CUDB is 40 (CS5):

```
# iptables -t mangle -A OUTPUT -p tcp --dport 389 -j DSCP --set-dscp 40
```
 - Default DiffServ Code Point for SNMP traps is 32 (CS4):

```
# iptables -t mangle -A OUTPUT -p udp --dport 162 -j DSCP --set-dscp 32
```

To make these rules persistent, do the following:

2. Edit the `/etc/puppet/modules/iptables/templates/iptables_rules_custom.erb`, and insert the following lines:

```
*mangle
-A OUTPUT -j DSCP --set-dscp 16
-A OUTPUT -p tcp -m tcp --dport 5555
-j DSCP --set-dscp 40
-A OUTPUT -p udp -m udp --dport 162
-j DSCP --set-dscp 40
COMMIT
```

3. Save the file.



5.7 Configuring SSL

The SSL administrative procedures described in this section are designated for SSL communication through the following external interfaces:

- tomcat-server (port 8383)
- CAI3G (port 8181)
- CAI3G (test port 8989)
- CAI (port 3301)
- CLI (port 8992)
- cudb-block-proxy (ports 8099,8994)

5.7.1 General Information

The following guide, throughout these subsections, is a recommendation on how to configure SSL to work on CAI3G (HTTPS/SOAP), CAI (Telnet), CUDB Block Proxy (JMX over RMI), and Dynamic Activation GUI (HTTPS).

Since the Private Key Infrastructure (PKI), and SSL setups vary from one customer installation to another, this instruction is to be seen as a verified guide on how it can be done.

5.7.2 Remarks and Deviations

The following section contains information about critical points, deviations, and remarks regarding the different SSL configurations.

5.7.2.1 Tomcat-server Module

Only one private key and certificate pair is verified for this instruction.

5.7.2.2 NBIA Module

A limited jetty SSL configuration is used that permits only one ssl certificate in the java keystore. If more than one private key, and certificate pair is wanted in the java keystore file, break out the certificates.

5.7.2.3 Cudb-block-proxy

Only one private key and certificate pair is verified for this instruction.



5.7.3 Configuration Instruction

On a high-level, a complete SSL configuration flow contains the following:

- Prerequisite configuration:
 - Create keystore file.
 - Create a Certificate Signing Request (CSR) request.
 - Get Certificate Authority (CA) certificate and signed server certificate from CA organization.
 - Import CA certificate and then the signed server certificate into the keystore file.
 - Populate the system with the keystore files and SSL settings.
 - Activate the SSL settings.

Note: Two keystore files are maintained: one is traffic keystore for NBIA module, and the other is OAM keystore for other modules.

Due to several modules use legacy SSL configuration parameters without the `keyalias` parameter, only one key – one certificate pair can be used for each module.

- Module/protocol specific configuration for every protocol:
 - NBIA module (port 8181), NBIA module (test port 8989)
 - tomcat-server module (port 8383)
 - cudb-block-proxy module (ports 8099,8994)
 - CLI module (port 8992), see Section 6.2.1 on page 133.

Note: The individual module/protocol configuration items are independent of each other.

A node-by-node activation is required before an individual configuration takes effect in system. Therefore, if multiple modules require SSL communication, it recommends configuring items for all those modules and then performing the activation and verification.

5.7.3.1 Prerequisite Configuration for NBIA Module

1. Generate a new private key and add it to the `pgexternal_trf_keystore.jks` keystore.

The new key is to be used for all external communication.



Note: Below is an example using example-specific values, such as `-dname` and similar. Choose your own parameters according to real situation.

```
$ sudo -u actadm /usr/java/latest/bin/keytool
-genkey -keyalg RSA -alias pgexternaltrf -keystore
/home/bootloader/ssl/pgexternal_trf_keystore.jks
-validity 3600 -keysize 2048 -dname 'CN=eg8-vip-trf.
ete.ka.sw.ericsson.se, OU=IT, O=Ericsson, L=Unknown,
ST=Stockholm, C=SE'
```

When prompted for password set keystore password.

2. Generate a CSR.

```
$ sudo -u actadm /usr/java/latest/bin/keytool -keystore
/home/bootloader/ssl/pgexternal_trf_keystore.jks
-certreq -alias pgexternaltrf -keyalg RSA -file
/home/bootloader/ssl/pgexternaltrf.csr
```

3. Verify the creation of the CSR.

```
$ cat /home/bootloader/ssl/pgexternaltrf.csr
```

Output:

```
-----BEGIN NEW CERTIFICATE REQUEST-----
/.../
-----END NEW CERTIFICATE REQUEST-----
```

4. Send the CSR to your Certificate Authority (CA). Your CA will send you back a number of intermediate certificate and signed server certificate files.

Note: The number of certificates are differ due to CA vendor.

A certificate file looks like the following:

```
-----BEGIN CERTIFICATE-----
[encoded data]
-----END CERTIFICATE-----
```

5. Save all those files received from CA to `/home/bootloader/ssl/`.
6. Use the `keytool` to import the certificate files into the keystore, one by one.

Note: The following example imports three certificates from CA.

- Import Root CA certificate file `verisignRoot.crt`:

```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/pgexternal_trf_keyst
ore.jks -file /home/bootloader/ssl/verisignRoot.crt
-alias verisignrootca
```



Enter keystore password:

Certificate was added to keystore

- Import Intermediate CA/CAs example file `intermediate.crt`:

```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/pgexternal_trf_keyst
ore.jks -file /home/bootloader/ssl/intermediate.crt
-alias intermediateca
```

Enter keystore password:

Certificate was added to keystore

- Import Server certificate file `pgexternaltrf.crt`:

```
$ sudo -u actadm /usr/java/latest/bin/keytool
-importcert -keystore /home/bootloader/ssl/pgexterna
l_trf_keystore.jks -file /home/bootloader/ssl/pgext
ernaltrf.crt -alias pgexternaltrf
```

Enter keystore password:

Certificate was added to keystore

Note: If the following error is shown:

```
keytool error: java.security.cert.CertificateException:
java.io.IOException: Incomplete data
```

To solve the problem, change the certificate file from `.crt` to a `.der` file and then import. For example:

```
$ sudo -u actadm openssl x509 -outform der -in
pgexternaltrf.crt -out pgexternaltrf.der
```

```
$ sudo -u actadm /usr/java/latest/bin/keytool
-importcert -keystore /home/bootloader
/ssl/pgexternal_trf_keystore.jks -file
/home/bootloader/ssl/pgexternaltrf.der -alias
pgexternaltrf
```

5.7.3.2

Prerequisite Configuration for Other Modules

For modules other than NBIA module, the prerequisite configuration is the same as NBIA, except those modules use keystore `pgexternal_oam_keystore.jks`.

An example is given as follows. The differences in commands are using “oam” keystore and certificate files (NBIA module uses “trf” instead). For more instruction details, see Section 5.7.3.1 on page 74.



1.

```
$ sudo -u actadm /usr/java/latest/bin/keytool
-genkey -keyalg RSA -alias pgexternaloam -keystore
/home/bootloader/ssl/pgexternal_oam_keystore.jks
-validity 3600 -keysize 2048 -dname 'CN=eg8-vip-trf.
ete.ka.sw.ericsson.se, OU=IT, O=Ericsson, L=Unknown,
ST=Stockholm, C=SE'
```

When prompted for password set keystore password.

2.

```
$ sudo -u actadm /usr/java/latest/bin/keytool -keystore
/home/bootloader/ssl/pgexternal_oam_keystore.jks
-certreq -alias pgexternaloam -keyalg RSA -file
/home/bootloader/ssl/pgexternaloam.csr
```
3.

```
$ cat pgexternal.csr
```
4. Send the CSR to your CA and get certificate files back. In this example, there are three files as follows:
 - Root CA certificate: verisignRoot.crt
 - Intermediate certificate: intermediate.crt
 - Service certificate: pgexternaloam.crt
5. Save the certificate files to /home/bootloader/ssl/.
6. Import those certificate files to the keystore one by one.
 - ```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/pgexternal_oam_keyst
ore.jks -file /home/bootloader/ssl/verisignRoot.crt
-alias verisignrootca
```

Enter keystore password:

Certificate was added to keystore
  - ```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/pgexternal_oam_keyst
ore.jks -file /home/bootloader/ssl/intermediate.crt
-alias intermediateca
```

Enter keystore password:

Certificate was added to keystore
 - Import Server certificate file pgexternaltrf.crt:

```
$ sudo -u actadm /usr/java/latest/bin/keytool
-importcert -keystore /home/bootloader/ssl/pgexterna
l_oam_keystore.jks -file /home/bootloader/ssl/pgext
ernaloam.crt -alias pgexternaloam
```



Enter keystore password:

Certificate was added to keystore

Note: If `java.io.IOException: Incomplete data error` is detected, change the certificate files from `.crt` to `.der` files before importing them. For example:

```
$ sudo -u actadm openssl x509 -outform der -in  
pgexternaloam.crt -out pgexternaloam.der
```

5.7.3.3 NBIA Module SSL Configuration and Activation

1. Set keystore for NBIA module:

```
$ bootloader.py config set --parameter @NBIA_KEYSTOREE  
XT@ --value /home/bootloader/ssl/pgexternal_trf_keys  
tore.jks
```

2. Encrypt the clear text password:

```
$ java -cp /usr/java/latest/lib/missioncontro  
l/plugins/org.eclipse.jetty.util_<version>.jar  
org.eclipse.jetty.util.security.Password <password>
```

Example Output:

```
OBF:1u2a1toalw8v1tok1u301u2a1toalw8v1tok1u30  
MD5:f6fdffe48c908deb0f4c3bd36c032e72
```

3. Set keystore password for the NBIA module:

```
$ bootloader.py config set --parameter @NBIA_SSLKEYPA  
SSWORD@ --value OBF:1u2a1toalw8v1tok1u301u2a1toalw8v  
1tok1u30
```

```
$ bootloader.py config set --parameter @NBIA_SSLPASSWOR  
D@ --value OBF:1u2a1toalw8v1tok1u301u2a1toalw8v1tok1u30
```

4. Verify that all parameters have been set:

```
$ bootloader.py config list
```

5. Perform an activation for the configured parameters to take effect on all modules on the system, one by one:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: Repeat the command for each node in the cluster.

6. Verify the certificate from an external node:

- NBIA CAI3G, NBIA module:

```
$ openssl s_client -connect <VIP-TRF-IP>:8181
-showcerts -tls1
```

Output:

```
CONNECTED(00000003)
/.../
<Certificate output>
```

- NBIA CAI3G Test Ports, NBIA module:

```
$ openssl s_client -connect <VIP-TRF-IP>:8989
-showcerts -tls1
```

Output:

```
CONNECTED(00000003)
/.../
<Certificate output>
```

5.7.3.4

Tomcat-server Module SSL Configuration and Activation

1. Set keystore for tomcat-server module:

```
$ bootloader.py config set --parameter @TOMCAT_KEYSTORE
_FILE@ --value /home/bootloader/ssl/pgexternal_oam_k
eystore.jks
```

2. Set the keystore password:



```
$ bootloader.py config set --parameter @TOMCAT_KEYSTORE
_PASSWORD@ --value <password>
```

3. Set the private alias:

```
$ bootloader.py config set --parameter @AUTHN_PRIVATE_A
LIAS@ --value pgexternaloam
```

4. Set the public alias:

```
$ bootloader.py config set --parameter @AUTHN_PUBLIC_AL
IAS@ --value pgexternaloam
```

5. Perform an activation for the configured parameters to take effect on all modules on the system, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: Repeat the command for each node in the cluster.

6. Verify the certificate from an external node:

```
$ openssl s_client -connect <VIP-OAM-IP>:8383 -showcerts -tls1
```

Expected response example:

```
CONNECTED(00000003)
depth=2 /C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - For a
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
0 s:/C=SE/ST=Stockholm/L=Stockholm/O=Ericsson/OU=IT/CN=eg8-vip-oam.ete.ka.sw.ericsson.se
i:/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure Serv
-----BEGIN CERTIFICATE-----
MIIGCzCCBPogAwIBAgIQXEKFIasv3kVZjna6Ms73NjANBgkqhkiG9w0BAQsFADB+
MQswCQYDVQQGEwJVUzEdMBsGA1UEChMUU3ltYW50ZWMMgQ29ycG9yYXRpb24xHzAd
BgNVBAsTFln5bWFudGVjIFRydXN0IE5ldHdvcmxLzAtBgNVBAMTJlN5bWFudGVj
IENsYXNzIDMgU2VjdXJlIFNlcnZlciBDQSAtIEc0MB4XDTE3MDEwOTAwMDAwMFoX
DTE4MDEwMDIzNTk1OVowgYExCzAJBgNVBAYTA1NFMRIwEAYDVQQIDAlTdG9ja2hv
bG0xEjAQBgNVBACMCVN0b2NraG9sbTERMA8GA1UECgwIRXJpY3Nzb24xCzAJBgNV
-----
```



5.7.3.5

Cudb-block-proxy Module SSL Configuration and Activation

1. Set keystore for cudb-block-proxy module:

```
$ bootloader.py config set --parameter @BLOCK_PROXY_SSL
LCERT_STORE@ --value /home/bootloader/ssl/pgexternal_o
am_keystore.jks
```

2. Activate SSL of cudb-block-proxy module:

```
$ bootloader.py config set --parameter @BLOCK_PROXY_JM
XREMOTE_SSL@ --value true
```

```
$ encrypt.sh '<clear_text_password>'
```

Note: The password needs to be inside the single quotation marks (').

Output:

```
pwqrl5Un3PemL4v88evQ+Q==
```

```
$ bootloader.py config set --parameter @BLOCK_PROXY_SSL
CERT_STORE_PASSWORD@ --value pwqrl5Un3PemL4v88evQ+Q==
```

3. Perform an activation for the configured parameters to take effect on all modules on the system, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: Repeat the command for each node in the cluster.

4. Verify the certificate from an external node:

```
$ openssl s_client -connect <VIP-OAM-IP>:8994 -showcerts -tls1
```

Example Output:

```
CONNECTED(00000003)
/.../
<Certificate output>
```



5.7.3.6 CAI SSL Configuration and Activation

The CAI SERVER2 with port 3301 is run with SSL by default. To change the SSL configuration for CAI SERVER2, run the following commands:

1. `$ bootloader.py config set --parameter @CAI_SERVER_2_ENABLED_SSL@ --value <boolean value>`

Set *<boolean value>* to `true` to enable the SSL configuration for CAI SERVER2 with port 3301.

Set *<boolean value>* to `false` to disable the SSL configuration for CAI SERVER2 with port 3301.

2. Perform an activation for the configured parameters to take effect on all modules on the system, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: Repeat the command for each node in the cluster.

3. Verify the certificate from an external node:

```
$ openssl s_client -connect <VIP-OAM-IP>:3301 -showcerts -tls1
```

Output:

```
CONNECTED(00000003)
/.../
<Certificate output>
```

5.7.3.7 General Cleanup

Backup all SSL working files, including the SSL keystore passwords, from the system to a secure storage area.

Files to be backed up are the:

- Keystore files



- CSR request file
- CA authority certificate files

5.8 Configuring Authentication Rules of GUI Users

This section describes how to configure authentication rules for GUI users, including:

- Password rules that must be fulfilled when setting a password for GUI users.
- Input parameters of Brute Force Protection Algorithm.

5.8.1 Brute Force Protection Algorithm

Brute Force Protection Algorithm is applied on the default `admin` GUI user because this user cannot be permanently locked out from system.

Once the `AUTHN_MAX_LOGIN_ATTEMPTS` is reached for the default `admin` GUI user, the user is locked out for a period. The Brute Force Protection Algorithm determines the locking period as follows:

- When the `admin` user is locked for the 1st time, the locking period is `AUTHN_BRUTE_FORCE_INITIAL_LOCK_PERIOD` (5 seconds by default).
- From the 2nd time, the locking period is doubled each time, until the `AUTHN_BRUTE_FORCE_MAX_LOCK_PERIOD` is reached. For example:
 - 2nd locking period is 10 seconds.
 - 3rd locking period is 20 seconds.
 - 4th locking period is 40 seconds, and so on.
- Later on, every time when the user is locked out, the locking period is `AUTHN_BRUTE_FORCE_MAX_LOCK_PERIOD` (15 minutes by default).

5.8.2 View Authentication Parameters

To list all the configuration information for the system, run the following command from node-1:

```
$ bootloader.py config list --show_all
```

Then find the authentication parameters in `tomcat` section.

Table 20 lists all the parameters for authentication configuration.



Table 20 Authentication Parameters

Parameter	Description	Type	
AUTHN_MIN_PASSWD_LENGTH	Minimum password length ⁽¹⁾	Integer	12
AUTHN_MIN_DIGIT_CHARS	Minimum number of digits [0-9] within password	Integer	1
AUTHN_MIN_UPPER_CASE_CHARS	Minimum number of upper case characters [A-Z] within password	Integer	1
AUTHN_MIN_LOWER_CASE_CHARS	Minimum number of lower case characters [a-z] within password	Integer	1
AUTHN_MIN_SPECIAL_CHARS	Minimum number of special characters [!"#\$%&'()*+,-./:;<=>?@[\\]^_`{ }~] within password	Integer	1
AUTHN_MAX_DAYS_TO_REMEMBER	Number of days that a password cannot be reused	Integer	30
AUTHN_MAX_DAYS_TO_LIVE	Maximum days before a password needs to be changed	Integer	90
AUTHN_MAX_LOGIN_ATTEMPTS	Maximum number of failed login attempts before the user is locked out	Integer	3
AUTHN_BRUTE_FORCE_INITIAL_LOCK_PERIOD	Initial locking period (in milliseconds) of default <code>admin</code> GUI user after reaching "AUTHN_MAX_LOGIN_ATTEMPTS"	Integer	5000
AUTHN_BRUTE_FORCE_MAX_LOCK_PERIOD	Maximum locking period (in milliseconds) for the default <code>admin</code> GUI user.	Integer	90000

(1) The password length must consist of minimum 12 characters.

5.8.3 Modify an Authentication Parameter

To modify the default values of these parameters, do the following from node-1:

1. Run the `bootloader.py` command to set a new value for a parameter.

```
$ bootloader.py config set --parameter <parameterName>
--value <newValue>
```

The new value is saved in the `user_config` file.

Example of setting a new value for `AUTHN_MAX_DAYS_TO_REMEMBER` as 30:



```
$ bootloader.py config set --parameter @AUTHN_MAX_DAYS_TO_REMEMBER@
INFO - *** @AUTHN_MAX_DAYS_TO_REMEMBER@=30 inserted in user config
INFO - *** To activate the changed parameter, execute "bootloader.py node activate --host <hostname>"
```

Check `user_config` for verification.

```
$ cat /home/bootloader/config/user_config
@AUTHN_MAX_DAYS_TO_REMEMBER@=30
```

2. Perform an activation on each node, one by one, for the configured parameters to take effect on all modules on the system.

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the node that is to be activated.

Note: The new value will take effect after executed the command `bootloader node activate`.

GUI users can keep using their current password until expiration.

5.8.4 Restore an Authentication Parameter

A parameter is restored to default value by removing the modified value from the `user_config` file.

To restore a parameter, do the following from node-1:

1. Run the `bootloader.py` command to remove the modified value.

```
$ bootloader.py config remove --parameter <parameterName>
```

Example of restoring `AUTHN_MAX_DAYS_TO_REMEMBER` to default value:

```
$ bootloader.py config remove --parameter @AUTHN_MAX_DAYS_TO_REMEMBER@
INFO - *** Removed @AUTHN_MAX_DAYS_TO_REMEMBER@ from user config
INFO - *** To activate the changed parameter, execute "bootloader.py node activate --host <hostname>"
```

2. Perform an activation on each node, one by one, for the configured parameters to take effect on all modules on the system.



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

Note: The restored value will take effect after executed the command `bootloader node activate`.

GUI users can keep using their current password until expiration.

5.9 License Keys Administration

This section holds information on how to manually request and install a new license key and how to perform a rollback.

Note: There is no downtime when a new license is installed or a rollback is performed. The Sentinel server is accessed by Dynamic Activation once a day which means that Dynamic Activation has all licenses in cache. To speed up the process, it is possible to trigger an update through the Dynamic Activation GUI. Also, provisioning to the Dynamic Activation is not required to be stopped.

5.9.1 Requesting New License Key

Do the following as user `root` to request the new license key:

1. Login to the master VM (with type name node-1), where the Sentinel license server resides.
2. Obtain the license code from node-1 and node-2, where the license server is installed:

```
# cd /opt/sentinel/bin
```

```
# ./echoid
```

```
Sentinel RMS Development Kit 8.6.2.0053 Host Locking Code Information Utility  
Copyright (C) 2014 SafeNet, Inc.
```

```
Locking Code 1 : 2008-*1MS LHEN 9GMR X8EQ  
Locking Code 1 (Old Style) : 2008-BA44D
```



Note: The `echoid` command needs to be run, as user `root`, in the `/opt/sentinel/bin` directory. Otherwise the license provided in Ericsson License Information System (ELIS) will not work.

3. Provide all Locking Code for the ELIS and get the license file.

```
13 FAT1022833/5 Ni LONG NORMAL NETWORK EXCL 100000 INFINITE_KEYS
15 NOV 2013 11 NOV 2014 NO_SHR SLM_CODE 1 NON_COMMUTER NO_GRACE NO
CL_ND_LCK NON_REDUNDANT AB99F2008,FB97E2008 NO_HLD 5 M2M_Start,_
PS T,fzJ:wrGlhYlTlWOoaZ3VyN5wPB1aJd4HVM505BvjfWZAcemFO1DYtPLY:y90y
FWRupieI93p#AID=fe665bf5-a24a-4c3c-910b-e882f41cb146
```

Example 10 A License File

5.9.2 Installing License Key

To install a new license key, follow the procedure:

1. Back up the existing licenses if there are any. See Section 5.9.3 on page 88.
2. Set the environment variable `LSHOST` to the license server:

```
# export LSHOST=localhost
```

3. Import the licenses by using a licence file received from ELIS. See Section 5.9.1 on page 86.

The license can be imported either from the file directly, or from the string found in the file.

- **Import a license file directly**

```
# /opt/sentinel/bin/lslic -F <file_name>
```

Example printout

```
Sentinel RMS Development Kit 8.6.2.0053 License Addition/Deletion Utility
Copyright (C) 2014 SafeNet, Inc.
```

```
License code "13 FAT1022833/5 Ni LONG NORMAL NETWORK EXCL 100000 INFINITE_KEYS 15
NOV 2013 11 NOV 2014
NO_SHR SLM_CODE 1 NON_COMMUTER NO_GRACE NO_OVERDRAFT CL_ND_LCK NON_REDUNDANT
AB99F2008,FB97E2008 NO_HLD 5
M2M_Start,_PS T,fzJ:wrGlhYlTlWOoaZ3VyN5wPB1aJd4HVM505BvjfWZAcemFO1DYtPLY:
y90yLgTU2Vw2Z7C1xFWRupieI93p$AID=
fe665bf5-a24a-4c3c-910b-e882f41cb146" added on vip_internal
```

- **Import from a string in the license file**
 - a Copy the entire text in the license file as a license string.



Note: The license string cannot contain any word-wrapping and the whole string needs to be on a single line for Sentinel to be able to read it.

- b Import the licences from the license string.

```
# /opt/sentinel/bin/lslic -A "<license_string>"
```

Example printout

```
License Addition/Deletion Utility
Copyright (C) 2008 SafeNet, Inc.
License code
13 FAT1022833/5 Ni LONG NORMAL NETWORK EXCL 100000 INFINITE_KEYS
 15 NOV 2013 11 NOV 2014 NO_SHR_SLM_CODE 1 NON_COMMUTER NO_GRACE NO_OVERDRAFT
CL_ND_LCK_NON_REDUNDANT AB99F2008,FB97E2008 NO_HLD 5 M2M_Start,_
PS`T,fzJ:wrGlhYlTlWOoaZ3VyN5wPB1aJd4HVM505BvjfWZAcemFO1DYYtPLY:y90yLgTU2Vw2Z7C1x
FWRupieI93p$AID=fe665bf5-a24a-4c3c-910b-e882f41cb146
```

4. Restart the license server:

```
# 3ppmon stoplserv
```

```
# 3ppmon startlserv
```

5. Verify on the localhost, node-1, that the license is installed:

```
# /opt/sentinel/bin/lsmmon localhost
```

Note: Make sure that the response does not contain:

```
There is no license in the server
```

6. Repeat Step 1 to Step 5 on node-2.
7. For the new license to take effect, an update on the Dynamic Activation applications needs to be performed in the Dynamic Activation GUI.

Login to the Dynamic Activation web GUI. For detailed information, refer to *User Guide for Resource Activation*, Reference [7].

Go to the **System > Licenses** tab.

Click the update arrows for the appropriate Dynamic Activation application to load the license to the system.

8. If the new license is replacing a license, follow Section 5.9.5 on page 89 to remove the old license.

5.9.3

Back up License Keys

To back up license keys, do the following on both node-1 and node-2:

1. Login as user `root`, and create a backup folder.



```
# mkdir /home/actadm/licenses/backup
```

2. Back up the existing license.

```
# cp /home/actadm/licenses/lservrc /home/actadm/licenses/backup
```

5.9.4 Restoring a Backup of License Keys

To restore a backup of a license key, follow this procedure:

1. Login as user `root`, and stop the Sentinel server on node-1 and node-2:

```
# 3ppmon stoplserv
```

2. On node-1 and node-2, copy the backed up file `lservrc` from directory `/home/actadm/licenses/backup` to the original directory, `/home/actadm/licenses/`:

```
# cp /home/actadm/licenses/backup/lservrc /home/actadm/licenses/
```

3. Start the Sentinel server on node-1 and node-2:

```
# 3ppmon startlserv
```

4. For this procedure to take effect, an update on the Dynamic Activation applications needs to be performed in the Dynamic Activation GUI.

Login to the Dynamic Activation web GUI. For detailed information, refer to *User Guide for Resource Activation*, Reference [7].

Go to the **System > Licenses** tab.

Click the update arrows for the appropriate Dynamic Activation application to load the license on the system.

5.9.5 Removing a License Key

To remove a license key, follow the procedure:

1. Login to node-1 as `root` user, and set the environment variable `LSHOST` to the license server:

```
# export LSHOST=localhost
```

2. Print existing licenses.

```
# /opt/sentinel/bin/lsmmon localhost
```

Example printout



Sentinel RMS Development Kit 8.6.2.0053 Application Monitor

Copyright (C) 2014 SafeNet, Inc.

[Contacting Sentinel RMS Development Kit server on host "vip_internal"]

```
| - Feature Information
|   - Feature name           : "FAT1022833/1"
|   - Feature version        : ""
|
|   - License type           : "Normal License"
|   - License Version        : 0x08400000
|   - Commuter license allowed : NO
|   - Maximum concurrent user(s) : Unlimited
|   - Unreserved tokens in use : 0
|   - Available unreserved    : Unlimited
|   - Reserved tokens in use  : 0
|   - Available reserved     : 0
|   - Soft limit on users     : Unlimited
|   - License start date      : Fri Nov 15 00:00:00 2013
|   - Expiration date        : Tue Nov 11 23:59:59 2014
|   - Log encryption level    : 1
|   - Check time tamper      : Yes
|   - Application-server locking : Client-locked license.
|   - Client $1 locking code   : 2008-AB99F
|   - Client $2 locking code   : 2008-FB97E
|   - Additive/exclusive/aggregate : Exclusive license(overrides additive licenses).
|   - Token lifetime (heartbeat) : 300 secs (5 min(s))
|   - Public vendor information : SW_Basic,_PS
|   - Allowed on VM           : YES
|
| - License Information
|   - License Hash           : 1181152AD32E7EBE
|   - License type           : "Normal License"
|   - License Version        : 0x08400000
|   - License storage name    : /home/actadm/licenses/lserverc
|   - License status         : Active
|   - Commuter license allowed : NO
|   - Maximum concurrent user(s) : Unlimited
|   - Soft limit on users     : Unlimited
|   - License start date      : Fri Nov 15 00:00:00 2013
|   - Expiration date        : Tue Nov 11 23:59:59 2014
|   - Log encryption level    : 1
|   - Check time tamper      : Yes
|   - Application-server locking : Client-locked license.
|   - Client $1 locking code   : 2008-AB99F
|   - Client $2 locking code   : 2008-FB97E
|   - Additive/exclusive/aggregate : Exclusive license(overrides additive licenses).
|   - Token lifetime (heartbeat) : 300 secs (5 min(s))
|   - Allowed on VM           : YES
```

3. Set the environment variable `LSFORCEHOST` to the license server:

```
# export LSFORCEHOST=localhost
```

4. Remove the specific License Key:

```
# /opt/sentinel/bin/lslic -df "<Feature name>" "<Feature Version>" "<License Hash>"
```

Example

```
# /opt/sentinel/bin/lslic -df "FAT1022833/1" ""
"1181152AD32E7EBE"
```




```
Sentinel RMS Development Kit 8.6.2.0053 License
Addition/Deletion Utility
Copyright (C) 2014 SafeNet, Inc.
```

This will delete license(s) from the server, do you want to continue? (Y/N):

Press Y.

```
license FAT1022833/1 is deleted.
```

5. Restart the license server:

```
# 3ppmon stoplserv
```

```
# 3ppmon startlserv
```

6. Verify on the localhost, node-1, that a license is still installed:

```
# /opt/sentinel/bin/lsmon localhost
```

Note: Make sure that the response does not contain:

```
There is no license in the server
```

7. Repeat Step 1 to Step 6 on node-2.

8. Verify that the license server vip address contains licenses:

```
# /opt/sentinel/bin/lsmon vip_internal
```

Make sure that the response does not contain the text:

```
There is no license in the server
```

9. For this procedure to take effect, an update on the Dynamic Activation applications needs to be performed in the Dynamic Activation GUI.

Login to the Dynamic Activation web GUI. For detailed information, refer to *User Guide for Resource Activation*, Reference [7].

Go to the **System > Licenses** tab.

Click the update arrows to load the license to the system.

5.10 Cassandra Administration

This section outlines useful `nodetool` commands to administer the Cassandra database.



5.10.1 Cassandra Data Consistency

5.10.1.1 Cassandra Node Outage

A Dynamic Activation cluster consisting of three nodes, replication factor 3, and read/write on consistency level `QUORUM`, can only have one node down at a time.

- When a node goes down, Cassandra saves incoming writes for that node, in accordance to how the `max_hint_window_ms` parameter is configured (three hours by default). When the node is up and running again, the cluster starts replicating missed data.

During node downtime, the node is still able to get consistent reads and writes, though Ericsson recommends getting the node back up and running as soon as possible.

- If the downtime is longer then the `max_hint_window_ms` configuration, the node requires manual repairing after it is up and running again. For instructions, see Section 5.10.1.2 on page 92.

5.10.1.2 Repair Cassandra Node

When replacing a blade, or a node goes down for an extended period of time (more than three hours by default), inconsistency occurs among Cassandra nodes. To synchronize Cassandra data, the replaced or down node needs to be repaired manually.

To repair a Cassandra node:

1. Log on to the node to be repaired.
2. Run the repair command.

```
$ nodetool repair
```

Note: Repairing a Cassandra node is a time consuming task. Depending on how much data is available in Cassandra, the `nodetool repair` command can continue in background up to several hours.

5.10.2 Cassandra Performance Monitoring

The following commands can be used to monitor the Cassandra cluster:

- `nodetool status logconsolidation` - Displays information about the Cassandra cluster.
- `nodetool compactionstats` - Displays the total column and the total number of uncompressed bytes of `SSTables` being compacted.
- `nodetool cfstats` - Displays statistics on each table and keyspace.



- `nodetool netstats` - Displays statistics on network operations and connections within the Cassandra cluster.
- `nodetool tpstats` - Displays statistics about the number of active, pending, and completed tasks for each stage of Cassandra operations by thread pool.

For more information about `nodetool` commands, see <https://docs.datastax.com/en/cassandra/2.2/cassandra/tools/toolsNodetool.html>, Reference [16].

5.11 PC Keys

This section contains information on how to install a new PC key and how to perform a rollback.

Note: There is downtime of each node when a new key is installed or a rollback is performed as the packages in the application are redeployed. The Dynamic Activation reads the keys from the key directory every 5 seconds and redeploys when changes are detected. Provisioning is not required to be stopped when adding keys to one node at a time in a cluster, allowing time for each node to get back up.

5.11.1 Installing a PC Key

To install a PC key, follow the steps:

1. Transfer the PC key to a temporary folder and pack it into a `.tar.gz` file:

```
$ tar cvzf PCkey-<Feature_Name>-<Rev>.tar.gz <PcKey>
```

2. Copy the packed PC key file to the `/opt/ericsson/activation/bootloader/repository/` directory:

```
$ sudo -u actadm cp PCkey-<Feature_Name>-<Rev>.tar.gz  
/opt/ericsson/activation/bootloader/repository/
```

3. If upgrading an old PC key, repeat the following command for all nodes to uninstall the old PC key.

From node-1:

```
$ bootloader.py submodule delete -n PCkey-<Feature_Name>  
-p dve-application --host <hostname>
```

`<hostname>` is the hostname of the processing node on which the old PC key is to be uninstalled.

4. Repeat the following command for all nodes to install the PC key.

From node-1:



```
$ bootloader.py submodule add -n PCkey-<Feature_Name>-<Rev>.tar.gz -t key -p dve-application --host <hostname>
```

<hostname> is the hostname of the processing node on which the PC key is to be installed.

5. From node-1, run the following command for all nodes, one by one, to activate the PC key:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

The keys are active as soon as the application has been redeployed, no other means of activation are necessary unless stated by the PC documentation.

5.11.2 Rolling Back a PC Key

To roll back a PC key, follow the steps:

1. Repeat the following command for all nodes to remove the PC key.

From node-1:

```
$ bootloader.py submodule delete -n PCkey-<Feature_Name>-<Rev>.tar.gz -t key -p dve-application --host <hostname>
```

<hostname> is the hostname of the processing node on which the PC key is to be removed.

2. If the current PC key is to be replaced by a previous key, ensure that the previous packed PC key file exists in the `/opt/ericsson/activation/bootloader/repository/`, and reinstall the previous key.

From node-1:

```
$ bootloader.py submodule add -n PCkey-<Feature_Name>-<Rev>.tar.gz -t key -p dve-application --host <hostname>
```

<hostname> is the hostname of the processing node on which the PC key is to be reinstalled.



3. From node-1, run the following command, for all nodes, one by one, to activate the PC key:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

4. Remove the packed PC key file from the `/opt/ericsson/activation/bootloader/repository/` directory:

```
$ cd /opt/ericsson/activation/bootloader/repository/
```

```
$ sudo -u actadm rm PCkey-<Feature_Name>-<Rev>.tar.gz
```

The keys are deactivated as soon as the application has been redeployed, no other means of deactivation are necessary unless stated by the PC documentation.

5.12 Processing Log Admin Tool

The Processing Log Admin Tool provides maintenance functionality for exporting processing logs from the data storage. This tool is also used by crontab in the Dynamic Activation cluster. When necessary, to release data storage space, it removes log entries in the Cassandra database (starting with the oldest).

The Processing Log Admin Tool can be run locally on one of the nodes within Dynamic Activation. It can also be run from an external node.

5.12.1 Using Processing Log Admin Tool

5.12.1.1 Exporting Logs

The following are ways to export logs using the Processing Log Admin Tool.

- Logs can be exported as days in UTC using the `-e` option. Then all log entries stored during a day are exported where a day is defined as `YYYY-MM-DD 00:00:00.000UTC` to `YYYY-MM-DD 23:59:59.999UTC`.



Logs exported as a day are unsorted to optimize export speed for backup purposes.

- Logs can be exported for a given time interval using the `-et` option. Then all complete logs (a northbound request with all belonging southbound requests) stored during the time interval are exported. The time interval is provided in the system local timezone. The output is sorted by the northbound entry timestamp.
- Complete logs can be exported sorted with the latest log first, using the `-el` option. This option searches the data store backwards until a provided limit of logs are exported. When exporting logs, a path for storing the export files must be provided.

Logs can be exported to an internal disk or to an external disk that is mounted onto the server. Logs can also be exported to a local path on an external computer that has access to the cluster. The exported log files are compressed to a .gz file, to minimize disk use.

Note: During export, for every 100000 logged entries a compressed file will be created. It is important to take into account that when exporting huge amount of data, these files can fill up the target partition, and it can take a few hours for the export of provisioning data to finish.

5.12.1.1.1

Running Processing Log Admin Tool

Table 21 shows the arguments to use when exporting processing logs:

Table 21 Arguments for Processing Log Admin Tool

Argument	Description
<code>-e</code>	<p>Starts the export on a given day, written as: YYYY-MM-DD, where a day is defined as YYYY-MM-DD 00:00:00.000UTC to YYYY-MM-DD 23:59:59.999UTC. This needs to be combined with the <code>-p</code> argument.</p> <p>This export is time prioritized and unsorted.</p> <p>Example:</p> <pre>sudo -u actadm ./proclog-admin-tool.sh -e 2015-06-20 -p /mnt/exportdisk</pre>
<code>-el</code>	<p>Starts the export from the last log saved in the database, and exports the given amount of logs.</p> <p>Example:</p> <pre>sudo -u actadm ./proclog-admin-tool.sh -el 100 -p /mnt/exportdisk</pre>
<code>-et</code>	<p>Starts the export with a start- and an end-date, write the two dates as one parameter.</p> <p>This export is sorted, and not time prioritized.</p> <p>Example:</p> <pre>sudo -u actadm ./proclog-admin-tool.sh -et "2015-06-20 12:00:00 2015-06-23 13:00:00" -p /mnt/exportdisk</pre>



-h	Help, lists all the existing commands that are applicable, and how to use them.
-l	Lists all the days a log has been stored in the database.
-p	This is the location where the files are exported to.
-rc	Removes the oldest days from the database so that only the configured number of days remain.
-ro	Removes the oldest day from the database

Run the Processing Log Admin Tool from Linux:

For local use, the Processing Log Admin Tool executable shell script, `proclog-admin-tool.sh`, is located in the `/usr/local/pgngn/admin-tool-<version>/bin` directory.

The following example shows how to export the processing logs created during one specified hour:

Example 1:

```
$ cd /usr/local/pgngn/admin-tool-<version>/bin

$ sudo -u actadm ./proclog-admin-tool.sh -et "2014-06-20
12:00:00 2014-06-23 13:00:00" -p <export_dir_path>
```

5.12.2

Configuring Processing Log Admin Tool

To update the Processing Log Admin Tool configuration, follow the instructions in Section 4.2 on page 37.

Note: The configuration change is node specific.

Log information derived by using the Processing Log Admin Tool, is saved in the `/var/log/dve/proclog-admin-tool.log` file.

Table 22 Processing Log Admin Tool Properties

Property	Description
<code>Cassandra.IP</code>	The IP address of the node where Cassandra resides. Multiple addresses can be specified by using a comma as a delimiter, for example: <code>169.254.100.1,169.254.100.2,169.254.100.3</code>
<code>Cassandra.Port</code>	The port for the Cassandra instance. Default is 9042
<code>export.file.numLogEntries</code>	Number of log entries per .csv file when exporting. Default is 100000
<code>cassandra.keyspace</code>	The keyspace used when searching. Default is <code>logconsolidation</code>



Property	Description
<code>cassandra.consistencyLevel</code>	The number of Cassandra nodes that must reply to a save request before proceeding. Default is <code>QUORUM</code>
<code>cassandra.replication</code>	The number of proclog data copies that need to be saved. Default is 3

5.13 Daily Export of Processing Logs

This section describes the export procedures of processing log data, on all nodes, on a daily basis.

The recommendation is to run the export script once a day. The script exports the processing logs that were provisioned the previous day.

The export process can be executed with the following script and flags.

- For exporting logs relative to the UTC time zone:

```
$ sudo -u actadm /usr/local/pgngn/admin-tool-<version>/bin/proclog-admin-tool-auto-export.sh utc  
<export_dir_path>
```

- For exporting logs relative to the time zone configured on the server:

```
$ sudo -u actadm /usr/local/pgngn/admin-tool-<version>/bin/proclog-admin-tool-auto-export.sh serverTimeZone  
<export_dir_path>
```

The script can be added as a crontab job.

Note: This script should only be executed on one node.

By default this script is not added to crontab due to the limit disk space for processing logs on file.

The path to where this script is exporting the data is configurable, and must be specified as an argument when running the script. It can for example be an internal disk, or an external disk that is mounted onto the server. Logs can also be exported to a local path on an external computer that has access to the cluster. The exported log files are compressed to a `.gz` file, to minimize disk use.

5.14 Log File Maintenance

This section contains information on log file maintenance of different logs, processing, and other.



5.14.1 Removal of Old Processing Logs

The `proclog-cleanup.sh` script is used by ESA to remove old processing logs. It removes logs, starting with the oldest, when the disk space exceeds a specified limit within ESA. The script then removes logs until the limit of 70% disk usage has been reached.

The script can be executed manually with different settings.

5.14.2 Configuration of the Retained Amount of Processing log days

To configure the retained amount of processing log days, follow the step-list:

1. From node-1, run the following command to configure the retention value:

```
$ bootloader.py config set --parameter @CASSANDRA_NUMBER_OF_DAYS_STORED@ --value <retention_value>
```

Default retention value is 30.

2. From node-1, run the following command, for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

5.14.3 Configuration of Processing Logs

Dynamic Activation provides log level settings.

Table 23 Log Levels

Log Level	Value	Description
0	ERROR	Only logs access control related administrative operations, for example, create a user, create authority context.
1	WARN	<ul style="list-style-type: none"> • Logs defined in log level 0 • Northbound (MML/CAI3G, and so on) request/response from/to the BSS system • GUI provisioning log



Log Level	Value	Description
2 (Default)	INFO	<ul style="list-style-type: none">Logs defined in log level 1Southbound request/response from/to the NEGUI management log (for example, create an NE)
3 ⁽¹⁾	DEBUG	<ul style="list-style-type: none">Logs defined in log level 2JDV/DV/DS level BL processing log. The message flow between activation logicsInternal component level processing log. The message flow between internal components.Future processing log sent by activation logics. ⁽¹⁾
-	OFF	Used to turn off <code>procllog</code> .

(1) After configuration updates, Dynamic Activation will distribute the detail information between the components in the Dynamic Activation system. Since a great amount of log data size could influence on the provisioning traffic and log disk size, be cautious in selecting this level.

The Log level value can be set to one of five different values:

Log level 0 = ERROR

Log level 1 = WARN

Log level 2 = INFO

Log level 3 = DEBUG

To turn off `procllog` = OFF

5.14.3.1

Configuration Steps of Processing Logs for Southbound Interfaces

To configure the process log, run the following commands:

1. From node-1, run the following command to configure log level setting for the processing log:

```
$ bootloader.py config set --parameter @LOG_LEVEL@  
--value <log_level_value>
```

2. From node-1, run the following command, for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```



<hostname> is the hostname of the processing node that is to be activated.

5.14.3.2 Configuration Steps of Processing Logs for CAI3G and CAI Interfaces

To configure the processing log, run the following commands:

1. From node-1, run the following command to configure log level setting for the processing log:

```
$ bootloader.py config set --parameter @PROCLOG_LEVEL@
--value <log_level_value>
```

2. From node-1, run the following command, for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

5.14.3.3 Configuration Steps of Processing Logs for MML Interfaces

To configure the processing log, run the following commands:

1. From node-1, run the following command to configure log level setting for the processing log:

```
$ bootloader.py config set --parameter @MML_PROCLOG_LEVEL@
--value <log_level_value>
```

Note: When setting the log level to `ERROR`, the faults of MML command are not recorded in the processing log. For detailed information about MML faults, refer to *Layered HLR AUC Provisioning over MML*, Reference [15].

2. From node-1, run the following command, for all nodes, one by one, to activate the change:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

5.14.4 Configuration of Flat Processing Log Files

Flat processing log files share the log level settings with processing log , see Section 5.14.3 on page 99.

Table 24 shows the configurable parameters of flat processing log files.

Table 24 Configurable Parameters for Flat Processing Logs Files

Parameter	Description	Type	Default Value ⁽¹⁾
FLATLOG_ENABLE_FLATLOGGING	Enable or disable the flat processing log files.	Boolean	true
FLATLOG_PATH_NBIA	The location of the flat processing log file generated by nbia-module.	String	/var/log/dve/flat log/
FLATLOG_PATH_ARH	The location of the flat processing log file generated by arh-module.	String	/var/log/dve/flat log/
FLATLOG_PATH_DVE_APPLICATION	The location of the flat processing log files generated by dve-application module.	String	/var/log/dve/flat log/
FLATLOG_PATH_TOMCAT_SERVER	The location of the flat processing log file generated by tomcat-server module.	String	/var/log/dve/flat log/
FLATLOG_PATH_DVE_INBOUND_INTERFACES_APPLICATION	The location of the flat processing log file generated by dve-inbound-interfaces-application module.	String	/var/log/dve/flat log/
FLATLOG_TIMEINTERVAL	The time interval in minutes between two flat processing log files.	Integer Value range: inclusively from 1 to 5	1
FLATLOG_LIFECYCLE	The time duration in minutes of a flat processing log file exists.	Integer Value range: inclusively from 5 to 60	10



Parameter	Description	Type	Default Value ⁽¹⁾
FLATLOG_MAXFILESIZE	The maximum size of the flat processing log file.	String 1MB-20MB	2MB
FLATLOG_LOGGING_COLUMNS	<p>Customize the fields to be logged in the flat processing log files. Field names are separated by comma and are case sensitive.</p> <p>The fields include the following:</p> <p>LogType, RootLogId, SubLogId, TransactionId, Instance, User, Target, Operation, Status, Hostname, Protocol, StartTime, ExecuteTime, ResponseCode, FullRequest, FullResponse, ReplayLogId</p>	String	LogType, RootLogId, SubLogId, TransactionId, Instance, Operation, Status, User, Hostname, Protocol, Target, StartTime, ExecuteTime, ResponseCode

(1) If a invalid value has been configured, flat processing log will use the default value.

To configure the flat processing log file parameters, run the following commands:

1. From node-1, run the following command to modified the value of a parameter:

```
$ bootloader.py config set --parameter <flat_processing_log_file_parameter> --value <flat_processing_log_parameter_value>
```

For example:

```
$ bootloader.py config set --parameter @FLATLOG_TIMEINTERVAL@ --value 2
```

2. From node-1, run the following command, for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```



<hostname> is the hostname of the processing node that is to be activated.

5.14.5 Other Log Files

Apart from the processing logs, most other log files are maintained by the script `auto_erase.sh` that is located in `/opt/dve/bin`

The script handles all log files where this is mentioned in Section 2.3.2 on page 11. The script also handles the CLI Response files, `/var/dve/cli`, which might contain personal data. The script checks the time stamp on those logs and result files regularly through a cron job. All files older than the systems retention time are deleted. The default retention time is 30 days.

The script can also be executed manually with different settings, see the following use help text:

Use:

```
$ /opt/dve/bin/auto_erase.sh --help
```

```
Usage: auto_erase.sh [OPTION...]
```

```
The script auto_erase.sh handles certain log and response files.
```

```
System Administrators Guide for PG NGN for more information
```

```
Examples:
```

```
auto_erase.sh          # Clean log files that are older than 30 days
```

```
auto_erase.sh -days 60 # Clean log files that are older than 60 days, values between 1 and 3
```

```
Options:
```

```
-days    change the number of days when certain log files is deleted
```

5.15 Automatic Queue Cleanup for Resilient Activation

The sub-requests in NE Abstraction Queues can grow over time and fill to capacity. Dynamic Activation provides an automatic cleanup function to cancel obsolete expired sub-requests.

5.15.1 How the Cleanup Function Works

When the cleanup function is activated, Dynamic Activation performs a scheduled job once a day, to cancel all sub-requests that have been expired for a specified retention period. Those sub-requests are treated as failed ones, and final notifications are sent to the client if the corresponding asynchronous requests of those sub-requests are in `Executing` state.

Note: The cleanup function takes effect on all NE Abstraction Queues.

5.15.2 Configure the Cleanup Function

To configure the cleanup function for Resilient Activation, do as follows from node-1:



1. Specify the number of days to keep an expired sub-request.

```
$ bootloader.py config set --parameter @EXPIRED_REQUEST_RETENTION_PERIOD@ --value <number of days>
```

Note: The default *<number of days>* is 0, which means the cleanup function is inactivated by default.

2. Schedule a time (0–23) for when to start the cleanup on a daily basis.

```
$ bootloader.py config set --parameter @EXPIRED_REQUEST_CLEANER_START_HOUR@ --value <hour>
```

Note: The *<hour>* is 2 by default, and is specified in a local time.

3. Active all nodes one by one, to apply the configuration.

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname of a node>
```

5.16 Enabling and Disabling Hostname in CAI3G Error Details

It is possible to enable and disable hostname in CAI3G error details.

To disable hostname, execute the following commands:

1. From node-1, run the following command to disable hostname in CAI3G error detail:

```
$ bootloader.py config set --parameter @CAI3G_DISABLE_HOSTNAME_IN_ERRORDETAILS@ --value true
```

2. From node-1, run the following command for all nodes, one by one, to activate the change:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

To enable hostname, execute the following commands:

1. From node-1, run the following command to enable hostname in CAI3G error detail:

```
$ bootloader.py config set --parameter @CAI3G_DISABLE_HOSTNAME_IN_ERRORDETAILS@ --value false
```

2. From node-1, run the following command for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

5.17 Configuring CAI3G Max Sessions

The number of simultaneous CAI3G session IDs can be changed from the default 25000 to suit the customer needs, by use of the `bootloader.py` commands. This is changed with the following procedure:

1. From node-1, run the following command to enable configuration of Max Sessions:

```
$ bootloader.py config set --parameter @CAI3G_MAX_SESSION@ --value <MaxSession>
```




2. From node-1, run the following command for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

5.18 Configuring MDV Time-Out

Note: The MDV timeout indicates the maximum time during which the MDV response must be returned. If MDV doesn't respond during the time, a timeout error is sent to client.

The MDV timeout is 300(s) by default, but it can be set to a larger value when customer has a complicated MDV, by using the `bootloader.py` command:

1. From node-1, run the following command to specify the MDV timeout value:

```
$ bootloader.py config set --parameter @MDV_TIMEOUT@
--value <MDV Timeout>
```

2. From node-1, run the following command for all nodes (one by one) to activate the change:

Note: Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

5.19 Synchronization Among Clusters

A Dynamic Activation cluster has license counters for itself and its synchronized clusters. Each counter represents one cluster. The sum of those counters are synchronized among the clusters and shown in GUI.

Attention!

Do not change the name of a cluster that is synchronized with other clusters. Otherwise it causes license-counter-mismatch problems.

Note: The clusters synchronize license counters every hour. During the intervals, the sum of counters on a cluster can be different from one another.

Note: The hardware-specific licenses, for example Max CPUs and Max VMs, are not distributed among clusters and are specific for each cluster. For example, two synchronized clusters with 8 CPUs in total, then each cluster must have a 4-CPU capacity license. This is shown in the GUI as 4 CPUs for the system.

To be able to synchronize clusters, a setup needs to be performed on each cluster first. During setup, generation and exchange of SSH keys are performed between clusters since scripts for synchronization of license counters and configuration work without passwords. This needs to be done on each cluster.

The following sections contain information on how to generate and exchange SSH keys between Dynamic Activation clusters. The keys are stored in `/home/syncuser/.ssh`.

Configuration file is located in the `/home/syncuser/config` directory. This file contains the IP addresses to the clusters to synchronize with and is populated when running the script for exchanging SSH keys in Section 5.19.2 on page 109.

Figure 3 acts as a conceptual picture of cluster synchronization where Cluster-A indicates currently used provisioning cluster with correct configuration, while Cluster-B and Cluster-C act as newly installed clusters.

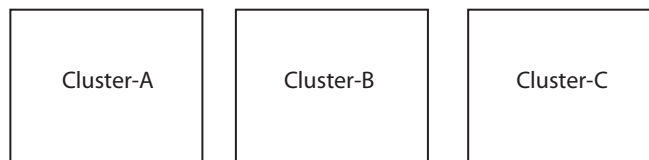


Figure 3 Conceptual Cluster Synchronization

5.19.1 Verification of Connections

Follow the instructions to check that the `ssh|sftp|scp` connection to the cluster, with which to exchange SSH keys, is working.

1. Enter the following command to check that the connection to the other cluster is working, and that the port for `ssh|sftp|scp` is open.



```
$ ssh actadm@<VIP-OAM cluster>
```

2. Check the output of the command.

- a If the following output is shown, the connection is working.

```
The authenticity of host '< VIP-OAM IP-address cluster>
(<VIP-OAM IP-address cluster 1>)' can't be established.
RSA key fingerprint is <autogenerated_key_fingerprint>.
Are you sure you want to continue connecting
(yes/no)?
```

Enter **no** to interrupt.

- b If the following output is shown, the connection is not working.

```
ssh: connect to host < VIP-OAM IP-address cluster>
port 22: Connection refused
```

5.19.2 Exchanging SSH Key between Clusters

This section contains information on how to generate and send SSH key from the newly installed clusters.

1. Make sure that all clusters are using the same software version.
2. Make sure that communication between the clusters is working, see Section 5.19.1 on page 108.
3. Change password for `syncuser` on each node of the cluster:

```
# passwd syncuser
```

4. Make sure that on all clusters that are synchronized, there are no old keys in the `/home/syncuser/.ssh` directory:

```
$ sudo -u syncuser rm -f /home/syncuser/.ssh/*
```

5. Log in on node-1 on the first cluster (Cluster-A according to Figure 3 to).
6. Run the setup script:

```
$ sudo -u syncuser /home/syncuser/scripts/sync_ssh_keys
.sh
```

A printout is shown:

```
Set up the communication for synchronization to one or
two other clusters. Before continuing, make sure the
IP addresses to the other clusters and the password for
syncuser are available.
```



Press any key to continue...

Press **Enter** to continue or interrupt with Ctrl+C if not all information specified above is available.

Specify the name of the cluster, VIP-OAM IP addresses for the clusters to synchronize with and username and password of a provisioning client user. Leave cluster 2 (Cluster C according to Figure 3) blank if only connect to one cluster.

Example:

When script is executed on Cluster-A, following values are used:

- Enter a name for this cluster:
Cluster-A
- Enter VIP-OAM IP-address of cluster 1:
VIP-OAM-IP of Cluster-B
- Enter VIP-OAM IP-address of cluster 2:
VIP-OAM-IP of Cluster-C

When script is executed on Cluster-B, following values are used:

- Enter a name for this cluster:
Cluster-B
- Enter VIP-OAM IP-address of cluster 1:
VIP-OAM-IP of Cluster-A
- Enter VIP-OAM IP-address of cluster 2:
VIP-OAM-IP of Cluster-C

When script is executed on Cluster-C, following values are used:

- Enter a name for this cluster:
Cluster-C
- Enter VIP-OAM IP-address of cluster 1:
VIP-OAM-IP of Cluster-A
- Enter VIP-OAM IP-address of cluster 2:
VIP-OAM-IP of Cluster-B

Enter a name for this cluster: **<name>**

Enter VIP-OAM IP-address of cluster 1: **<VIP-OAM
IP-address cluster 1>**

Enter VIP-OAM IP-address of cluster 2: **<VIP-OAM
IP-address cluster 2>**

Enter username for access: **<user name of a provisioning
client user>**

Enter password for access: **<password of the provisioning
client user>**

Encrypting password...

Password encrypted!

Current Cluster name and VIP-OAM IP addresses of other clusters:

THIS_CLUSTER_NAME<name>



```
CL1_VIP_OAM_IP <VIP-OAM IP-address cluster 1>
CL2_VIP_OAM_IP <VIP-OAM IP-address cluster 2>
JMX_USER admin <user name of a provisioning client user>
JMX_PASSWORD admin <password of the provisioning client
user>

Are the configurations above correct? (y/n) y

Transferring SSH keys to remote cluster <VIP-OAM
IP-address cluster 1> as syncuser

The authenticity of host '<VIP-OAM IP-address cluster 1>
(<VIP-OAM IP-address cluster 1>)' cannot be established.
RSA key fingerprint is <autogenerated_key_fingerprint>.
Are you sure you want to continue connecting (yes/no)?
```

Answer **Yes** to keep the connection to cluster 1.

Warning: Permanently added < VIP-OAM IP-address cluster 1>' (RSA) to the list of known hosts.

Specify SSH key:

Enter the password for user syncuser

Password: <syncuser_password>

authorized_keys 100% 400 0.4KB/s 00:00

Note: Password required in this step is the syncuser password set in Step 3.

Copying syncuser files to all nodes in cluster.

The following output is printed once for each node in the cluster:

```
The authenticity of host '<nodeId> <IP address>' cannot
be established.
ECDSA key fingerprint is <autogenerated_key_fingerprint>.
Are you sure you want to continue connecting (yes/no)?
```

Answer **Yes** for each node in the cluster.

Warning: Permanently added <nodeId,<IP address>' (ECDSA) to the list of known hosts.

syncuser@<nodeId>'s password:

Enter the password for user syncuser and press **Enter**



```
syncuser@<nodeId>'s password:
```

Enter the password for user `syncuser` and press **Enter**

```
Copying of syncuser files done.
```

7. Log in on node-1 on the next cluster (Cluster-B, according to Figure 3), and repeat Step 6.

8. Log on to each cluster and verify that no password is needed:

```
$ sudo -u syncuser ssh syncuser@<VIP-OAM_IP-address_cluster_1_or_2>
```

5.19.3 Forward SSH Key to New Cluster

This section contains information on how to generate and exchange SSH key when adding the second cluster (cluster2).

The following instructions are to be performed on the first cluster. On cluster2 that is added, perform the instructions specified in Section 5.19.2 on page 109.

1. Make sure that all clusters are using the same software version.
2. Make sure that communication to the new cluster is working, see Section 5.19.1 on page 108.
3. Log in to node-1.
4. Run the setup script:

```
$ cd /home/syncuser/scripts
```

```
$ sudo -u syncuser ./sync_ssh_keys.sh
```

5. A printout is shown:

```
Set up the communication for synchronization to one or
two other clusters. Before continuing, make sure the
IP addresses to the other clusters and the password for
syncuser are available. Press any key to continue.
```

Press **Enter** to continue or interrupt with Ctrl+C if not all information specified above is available.

6. The IP address of cluster 1 is shown:

```
Current VIP-OAM IP addresses of other clusters:CL1_VIP_
OAM_IP < VIP-OAM IP-address cluster 1> CL2_VIP_OAM_IP
```

```
Are the above IP addresses correct? (y/n)
```

Answer **n**



7. Enter the IP address for cluster 1 and the new IP address for cluster 2.

8. Forward the SSH key to cluster 2:

- a. The authenticity of host '*< VIP-OAM IP-address cluster 2> (<VIP-OAM IP-address cluster 2>)*' cannot be established. RSA key fingerprint is *<autogenerated_key_fingerprint>*. Are you sure you want to continue connecting (yes/no)?

Answer **yes** to keep the connection to cluster 2.

- b. Warning: Permanently added *< VIP-OAM IP-address cluster 2>* (RSA) to the list of known hosts.

Then the SSH key is sent to cluster 2.

5.19.4 Exchanging SSH Key When Replacing a Cluster

This section contains information on how to generate and exchange SSH key when replacing a cluster. Exchange of SSH key is needed since the scripts for synchronization cannot handle passwords.

This is only needed for synchronization of license counters and configuration and is to be performed on the cluster already available.

1. Make sure that all clusters are using the same software version.
2. Make sure that communication to the new cluster is working, see Section 5.19.1 on page 108.
3. Log in to node-1.
4. Run the setup script:

```
$ cd /home/syncuser/scripts
```

```
$ sudo -u syncuser ./sync_ssh_keys.sh
```

5. A printout is shown:

```
Set up the communication for synchronization to one or
two other clusters. Before continuing, make sure the
IP addresses to the other clusters and the password for
syncuser are available. Press any key to continue.
```

Press **Enter** to continue or interrupt with Ctrl+C if not all information specified above is available.

6. The current IP addresses of the clusters are shown:



```
Current VIP-OAM IP addresses of other clusters:CL1_VIP
_OAM_IP < VIP-OAM IP-address cluster 1> CL2_VIP_OAM_IP <
VIP-OAM IP-address cluster 2>
```

Are the above IP addresses correct? (y/n)

Answer **n**

7. Enter the IP addresses for the clusters. Change the address that is new.

8. Forward the SSH key to the new cluster:

- a. The authenticity of host '< VIP-OAM IP-address cluster "new cluster"> (<VIP-OAM IP-address cluster "new cluster">)' cannot be established. RSA key fingerprint is <autogenerated_key_fingerprint>. Are you sure you want to continue connecting (yes/no)?

Answer **yes** to keep the connection to the new cluster.

- b. Warning: Permanently added < VIP-OAM IP-address cluster "new cluster"> (RSA) to the list of known hosts.

Then the SSH key is sent to the new cluster.

5.19.5 Update Password for Cluster Synchronization

This section contains information on how to update the password for the cluster synchronization.

Attention!

Whenever the password of the provisioning client user, selected in Section 5.19.2 on page 109, is changed in the **Operation & Management >Access Control** GUI, the same password must be updated on the cluster for cluster synchronization.

Otherwise the synchronization breaks.

1. Run the setup script by using the “-changePassword” parameter.

```
$ cd /home/syncuser/scripts
```

```
$ sudo -u syncuser ./sync_ssh_keys.sh -changePassword
```

2. Confirm to update the password by entering **y** and then pressing **Enter**.



Do you want to update the password for syncing (y/n)

y

3. The following printout is displayed.

Enter the new password for access:

4. Enter the same password as that in the GUI, and then press **Enter**.

5.19.6 License Counters Synchronization

The script `sync_license.sh`, located in `/home/syncuser/scripts`, extracts license counters and synchronizes these with the other clusters. This script is run by a crontab job once every hour on each node in the cluster with `syncuser` authority and with an interval of five minutes. The time interval prevents the nodes from executing the job at the same time.

This means that the script starts on the first node at 00, continuing on the second node at 05, on the third at 10 and so on.

The temporary files containing the license counter that is transferred between the clusters are at `/home/syncuser`

The temporary file containing the configuration that is transferred between the clusters is at `/home/syncuser/scripts`

Log files for synchronization function are located in `/home/syncuser/logs`

These logs describe the license counters that were sent to and received from another cluster.

5.19.7 Remove Clusters from Synchronization

To remove clusters, the synchronization must be stopped and the synchronized counters must be cleaned.

Note: The same procedures must be performed on every cluster in the synchronization.

Stop the Synchronization

Perform the following procedure on each cluster in the synchronization:

1. Back up the property file for synchronization. The file is:

`/home/syncuser/config/sync.properties`

2. In the current property file (`/home/syncuser/config/sync.properties`), remove the address of one or more clusters that it no longer should synchronize with.



For example, when `cluster_1`, `cluster_2` and `cluster_3` are synchronized, and `cluster_3` is to remove from synchronization:

Change `cluster_1` properties to:

```
CL1_VIP_OAM_IP=cluster_2
```

```
CL2_VIP_OAM_IP=
```

Change `cluster_2` properties to:

```
CL1_VIP_OAM_IP=cluster_1
```

```
CL2_VIP_OAM_IP=
```

Change `cluster_3` properties to:

```
CL1_VIP_OAM_IP=
```

```
CL2_VIP_OAM_IP=
```

Clean the Counters

1. Get all cluster names by running the command multiple times, each time using a *cluster IP* of those clusters.

```
$ ssh <cluster IP> less /home/syncuser/config/sync.properties | grep THIS_CLUSTER_NAME
```

2. Perform the following procedure on node-1 for each cluster in the synchronization:

- a. Remove all pending imports of counters on the current cluster.

```
$ sudo -u syncuser rm /home/syncuser/logs/*.cc
```

- b. Clean the counters of other clusters that the current cluster should no longer synchronize with.

```
$ /opt/dve/tools/licenseclient/licenseclient.sh
deleteexternalusedcapacity -u=<provisioning
client user> -e=<provisioning client user password>
-cid=<cluster name>
```

Note: The *<cluster name>* was obtained in Step 1.

For example, when `cluster_1` ~ `cluster_3` are synchronized, and `cluster_3` is to remove from synchronization:

- On `cluster_1` and `cluster_2`, clean the counter by using *name of cluster_3*.
- On `cluster_3`, clean two counters by using *name of cluster_1* and *name of cluster_2*.



5.20 Dynamic Activation Application Custom Log4j Settings

By use of the `bootloader.py` command, see Section 4.1 on page 35, it is possible to set up logging with a custom log level for modules in the Dynamic Activation Application. These custom logs are redirected to a user-provided log output file, located in the `/var/log/dve/customlogs/` directory.

Applying new log settings is performed through the `bootloader.py deploy-log` command, and are persisted over both a reboot and activate of the system. The `bootloader logging deploy-log` command can be used to deploy custom logging for a sub-module on one host at a time, or on all hosts where the specific sub-module resides.

Caution!

This feature is used for debugging and troubleshooting. It should not be enabled on a live deployment. Depending on custom logging amount deployed, this may have severe impact on application performance.

The commands in Section 5.20.1 on page 117 through Section 5.20.3 on page 118, should all be executed from the first node in the cluster (node-1).

All the custom logs in `/var/log/dve/customlogs/` have a maximum size of 15 MB before they are rotated, and stores a maximum of 10 log files before the oldest logs are deleted.

5.20.1 Deploy Custom Log Settings

This section describes the procedure of deploying custom log settings on Dynamic Activation Application modules.

1. To deploy a custom log setting, execute the following command from node-1:

```
$ bootloader.py logging deploy-log -n <(sub)module name>
-j <jarfile> -l <loglevel> --host <ip/hostname/all> -o <log
file name>
```

Note: To list all the available jar-files for a specific sub-module, use the following command from node-1; `bootloader.py logging list -n <module-name>`.

2. Verify that the new settings are applied.

From node-1:

```
$ bootloader.py logging list -t log
```



Make sure the new log setting exists in the list.

The log output file can be found in the `/var/log/dve/customlogs/` directory, on the node where custom logging was enabled.

Note: The `-outputfile/-o` flag in `bootloader logging deploy-log` command, is optional. If no output filename is provided, the output file is named `<module name>.log`

The following shows an example of enabling debug on all the classes in `DVE-Core-2.359.jar`, contained in `dve-application` module. The settings are deployed to host `cluster-host-1`

The custom logging is stored in `/var/log/dve/customlogs/example.log`

```
$ bootloader.py logging deploy-log -n dve-application
-j DVE-Core-2.359.jar -l debug --host cluster-host-1 -o
example.log
```

Note: The host `cluster-host-1` can be replaced with `all` to enable debug on all the hosts in the cluster, where `dve-application` exists.

5.20.2 Undeploy Custom Log Settings

This section describes the procedure of undeploying activate log settings through the bootloader. This is performed by use of the `bootloader.py logging undeploy-log` command.

1. To undeploy a custom log setting, execute the following command from node-1:

```
$ bootloader.py logging undeploy-log -n <(sub)module
name> -j <jarfile> --host <ip/hostname/all> -o <log file
name>
```

2. Verify that the log settings were correctly removed, execute the following command from node-1:

```
$ bootloader.py logging list -t log
```

Make sure that the list no longer contains any custom settings.

The following shows an example of disabling the settings applied in Step 2:

```
$ bootloader.py logging undeploy-log -n dve-application
-j DVE-Core-2.359.jar --host cluster-host-1
```

Note: The output log is not removed during undeploy, and is saved on disk until manually removed. Some cleanup may be necessary when old debug logs are no longer needed.



5.20.3 List Active Custom Log Settings

This section describes how to list all the active custom log settings in the cluster.

1. To list all the active log settings in the cluster, execute the following command from node-1:

```
$ bootloader.py logging list -t log
```

2. To show a list of available JAR files in a specific sub-module, execute the following command from node-1:

```
$ bootloader.py logging list -n <(sub)module name>
```

5.21 Configuring External OpenID Connect Provider

This section describes how to configure Dynamic Activation to connect to external OpenID Connect Provider.

Table 25 lists all the configuration parameters for connecting to external OpenID Connect Provider configuration.

Table 25 Configuration Parameters for Connecting to External OpenID Connect Provider

Parameter	Description	Default Value	Occurrence
AUTHN_AUTHENTICATION_URI ⁽¹⁾	Authorization Endpoint URL of the OpenID Connect Provider	N/A	Mandatory
AUTHN_TOKEN_URI ⁽¹⁾	Token Endpoint URI of the OpenID Connect Provider	N/A	Mandatory
CORE_CLIENT_ID ⁽¹⁾	OAuth 2.0 client Identifier of Dynamic Activation defined at the Authorization Server	N/A	Mandatory
CORE_CLIENT_SECRET ⁽¹⁾	OAuth 2.0 client secret of Dynamic Activation defined at the Authorization Server	N/A	Mandatory
CORE_REDIRECT_URI	Redirection URI of Dynamic Activation to which the authentication response will be sent. Format: <code>https://<VIP-OAM-IP>:8383/oauth/v1/callback</code>	N/A	Mandatory
AUTHN_PUBLIC_ALIASES ⁽²⁾	The alias of certificate in the keystore of Dynamic Activation used to validate the ID Tokens signature received from OpenID Connect Provider	slr_server	Mandatory



Parameter	Description	Default Value	Occurrence
AUTHN_ISSUER ⁽¹⁾	Issuer Identifier for the OP that the RP is to send the Authentication Request to. Its value MUST be a URL using the https scheme.	N/A	Mandatory
AUTHN_LOGOUT_URI ⁽¹⁾	End Session Endpoint URI of the OpenID Connect Provider for RP-initiated logout	N/A	Mandatory
POST_LOGOUT_REDIRECT_URI	Redirection URI of Dynamic Activation to which end user's User Agent is redirected after a logout has been performed. Format: <code>https://<VIP-OAM-IP>:8383</code>	N/A	Mandatory

(1) The value of these parameters depends on the OpenID Connect Provider product.

(2) The alias name is defined during setting OpenID Connect Provider configuration in Dynamic Activation. See example in Section 5.21.2 on page 121.

5.21.1

Pre-request for External OpenID Connect Provider

To utilize OpenID Connect services for an end user, Dynamic Activation must be registered with external OpenID Connect Provider to provide information about itself and to obtain required information, including OAuth 2.0 client ID and client secret. In general, the following information of Dynamic Activation must be registered in external OpenID Connect Provider:

- Redirect URI to Dynamic Activation for authentication response
- Redirect URI to Dynamic Activation for post logout

Dynamic Activation supports authentication using Authorization Code Flow. Dynamic Activation uses `profile` scope to require specific sets of claims. All required claims must be returned to Dynamic Activation in `ID_Token`. Dynamic Activation requires the following standard claims to be returned in `ID_Token`:

- `given_name`
- `family_name`
- `preferred_username`
- `email`
- `phone_number`

The procedures of client registration and configuration of `ID_Token` as well as claims depend on the OpenID Connect Provider product. For more information about OpenID Connect specifications, refer to OpenID Connect Core 1.0 and OpenID Connect Session Management 1.0.



Dynamic Activation always checks the time of `iat` claim in `ID_Token`. To avoid invalid time in `iat`, Dynamic Activation and external OpenID Connect Provider must be time synchronized through NTP.

5.21.2 Setting External OpenID Connect Provider Configuration in Dynamic Activation

To set external OpenID Connect Provider configuration in Dynamic Activation:

1. Export the certificate for ID Token signature from external OpenID Connect Provider server.
2. Import the certificate file for ID Token signature validation into the keystore of Dynamic Activation.

For example, run the following commands on node-1 to import ID Token signature certificate from Step 1:

- a Upload the certificate file to directory `/home/bootloader/ssl/`.
- b Import the certificate file into keystore:

```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/keystore_tomcat.jks
-file /home/bootloader/ssl/id_token_signature.crt
-alias test
```

Enter keystore password:

Certificate was added to keystore

Note: The password for `keystore_tomcat.jks` can be found in `/user/local/pgngn/tomcat-server-<version>/config/oauth.properties`.

3. Export certificate file for SSL connection from external OpenID Connect Provider server.
4. Import certificate file for SSL connection into the keystore of Dynamic Activation.

For example, run the following command on node-1 to import SSL connection certificate from Step 3:

- a Upload the certificate file to directory `/home/bootloader/ssl/`.
- b Import SSL connection certificate file into keystore:

```
$ sudo -u actadm /usr/java/latest/bin/keytool -import
-keystore /home/bootloader/ssl/keystore_tomcat.jks
-file /home/bootloader/ssl/ssl_connection.crt -alias
tomcat
```



Enter keystore password:

Certificate was added to keystore

Note: The password for `keystore_tomcat.jks` can be found in `/user/local/pgngn/tomcat-server-<version>/config/oauth.properties`

5. Configure external OpenID Connect Provider parameters in Table 25 for Dynamic Activation.

For example, when integrating with ForgeRock OpenAM server, run the following commands as the `actadm` user on Dynamic Activation node-1 to add external OpenID Connect Provider configuration:

```
$ bootloader.py config set --parameter @AUTHN_AUTHENTICATION_URI@ --value https://openam.com:8383/openam/oauth2/authorize
```

```
$ bootloader.py config set --parameter @AUTHN_TOKEN_URI@ --value https://openam.com:8383/openam/oauth2/access_token
```

```
$ bootloader.py config set --parameter @CORE_CLIENT_ID@ --value edaAgent
```

```
$ bootloader.py config set --parameter @CORE_CLIENT_SECRET@ --value edaagent000
```

```
$ bootloader.py config set --parameter @CORE_REDIRECT_URI@ --value https://10.170.13.62:8383/oauth/v1/callback
```

```
$ bootloader.py config set --parameter @AUTHN_PUBLIC_ALIAS@ --value test
```

Note: The alias name is the same as the one in Step 2.

```
$ bootloader.py config set --parameter @AUTHN_ISSUER@ --value https://openam.com:8383/openam/oauth2
```

```
$ bootloader.py config set --parameter @POST_LOGOUT_REDIRECT_URI@ --value https://10.170.13.62:8383
```

```
$ bootloader.py config set --parameter @AUTHN_LOGOUT_URI@ --value https://openam.com:8383/openam/oauth2/connect/endSession
```

6. From node-1, run the following command to activate the modification for each node, one by one:

Note: Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter can only be used when no provisioning traffic is running.



```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

5.21.3 Removing External OpenID Connect Provider Configuration

To remove external OpenID Connect Provider configuration:

1. Roll back Ericsson certificate from Dynamic Activation.

Note: The following example removes alias of ID Token signature certificate and SSL certificate.

The password for `keystore_tomcat.jks` can be found in
 /user/local/pgngn/tomcat-server-<version>/config
 /oauth.properties.

```
$ sudo -u actadm /usr/java/latest/bin/keytool -delete
-alias test -keystore /home/bootloader/ssl/keystore_tomcat.jks -storepass <password>
```

```
$ sudo -u actadm /usr/java/latest/bin/keytool -delete
-alias tomcat -keystore /home/bootloader/ssl/keystore_tomcat.jks -storepass <password>
```

2. From node-1, run the following command to remove external OpenID Connect Provider configuration in the cluster.

```
$ bootloader.py config remove --parameter @AUTHN_AUTHENTICATION_URI@
```

```
$ bootloader.py config remove --parameter @AUTHN_TOKEN_URI@
```

```
$ bootloader.py config remove --parameter @CORE_CLIENT_ID@
```

```
$ bootloader.py config remove --parameter @CORE_CLIENT_SECRET@
```

```
$ bootloader.py config remove --parameter @CORE_REDIRECT_URI@
```

```
$ bootloader.py config remove --parameter @AUTHN_PUBLIC_ALIAS@
```

```
$ bootloader.py config remove --parameter @AUTHN_ISSUER@
```

```
$ bootloader.py config remove --parameter @POST_LOGOUT_REDIRECT_URI@
```

```
$ bootloader.py config remove --parameter @AUTHN_LOGOUT_URI@
```



3. From node-1, run the following command to activate the modification for each node, one by one:

Note: Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter can only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the node that is to be activated.

5.22 Asynchronous Request Handler Settings

5.22.1 Configuration of Asynchronous CAI3G 1.2 Interface

This section contains information on how the Asynchronous CAI3G 1.2 interface configuration works the first time it gets enabled and operated, as well as how to modify the parameters.

The first time the Asynchronous CAI3G 1.2 interface is deployed, the parameters in Table 26 are initiated with default values.

To modify the default values of these parameters, do the following on node-1:

1. Run the `bootloader.py` command to list the configuration information for the system:

```
$ bootloader.py config list --show_all
```

Then find the Asynchronous CAI3G 1.2 interface configuration in `arh-module`.

2. Run the `bootloader.py` command to modify the values of a parameter.

- a To set the new value, do the following:

The new value is saved in the `user_config` file.

```
$ bootloader.py config set --parameter <parameter>
--value <value>
```

Example of setting a new value for `ARHJOB_MAX_QUEUE_SIZE`:

```
$ bootloader.py config set --parameter @ARHJOB_MAX_QUEUE_SIZE@
--value 10000
```

```
INFO - *** @ARHJOB_MAX_QUEUE_SIZE@=10000 inserted in
user configuration
INFO - *** To activate the changed
parameter, execute "bootloader.py node activate
--host <hostname>" command
```



Check `user_config` for the changed parameter:

```
$ cat /home/bootloader/config/user_config
```

```
@ARHJOB_MAX_QUEUE_SIZE@=10000
```

- b To restore the default value, do the following:

The value being restored is removed from the `user_config` file.

```
$ bootloader.py config remove --parameter <parameter>
```

Example of restoring the value for `ARHJOB_MAX_QUEUE_SIZE`:

```
$ bootloader.py config remove --parameter
@ARHJOB_MAX_QUEUE_SIZE@
```

```
INFO - *** Removed @ARHJOB_MAX_QUEUE_SIZE@ from
user config INFO - *** To activate the changed
parameter, execute "bootloader.py node activate
--host <hostname>" command uration
```

3. From node-1, run the following command to activate the modification for each node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the processing node that is to be activated.

4. Verify the changes by logging in to the Asynchronous CAI3G 1.2 interface.

Note:

- The new value will take effect after executed the command `bootloader node activate`.

Table 26 Configuration Parameter

Parameter	Description	Type	Default Value
ARHJOB_MAX_QUEUE_SIZE	Maximum number of requests can be stored in Asynchronous Request Handler.	Integer	10,000,000



Parameter	Description	Type	Default Value
CAPACITY_CHECK_INTERVAL	The interval in millisecond of checking the maximum number of requests in Asynchronous request Queue.	Integer	60,000
ARHNOTIFICATION_MAX_RETRY_TIME	The maximum retry time in millisecond of re-sending the final failed notification.	Integer	3,600,000
ARHNOTIFICATION_RETRY_FACTOR	The retry factor of the interval for retrying the final failed notification.	Integer	2

5.23 Batch Handler Settings

This section contains information on how the batch handler configuration works the first time it gets enabled and operated, as well as how to modify the parameters.

The first time the batch handler is deployed, the parameters in Page 128 are initiated with default values.

To modify the default values of these parameters, do the following on node-1:

1. Run the `bootloader.py` command to list the configuration information for the system:

```
$ bootloader.py config list --show_all
```

Then find the batch handler configuration in `ema-batch-module`.

2. Run the `bootloader.py` command to modify the values of a parameter.
 - a To set the new value, do the following:

The new value is saved in the `user_config` file.

```
$ bootloader.py config set --parameter <parameter>
--value <value>
```

Example of setting a new value for `ACS_RETRY_TIMES`:

```
$ bootloader.py config set --parameter @ACS_RETRY_TIMES@
--value 3
```

```
INFO - *** @ACS_RETRY_TIMES@=3 inserted in user
configuration
INFO - *** To activate the changed parameter,
execute "bootloader.py node activate --host
<hostname>" command
```



Check `user_config` for the changed parameter:

```
$ cat /home/bootloader/config/user_config

@ACS_RETRY_TIMES@=3
```

- b To restore the default value, do the following:

The value being restored is removed from the `user_config` file.

```
$ bootloader.py config remove --parameter <parameter>
```

Example of restoring the value for `ACS_RETRY_TIMES`:

```
$ bootloader.py config remove --parameter
@ACS_RETRY_TIMES@

INFO - *** Removed @ACS_RETRY_TIMES@ from user
configuration
INFO - *** To activate the changed parameter,
execute "bootloader.py node activate --host
<hostname>" command
```

3. From node-1, run the following command to activate the modification for each node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the processing node that is to be activated.

4. Verify the changes by logging in to the Batch Handler.

Note:

- The new value will take effect after executed the command `bootloader node activate`.

*Table 27 Configurable Parameters of Batch Module*

Parameter	Description	Type	Default Value
ACS_RETRY_TIMES	The number of times an Akka Communication Service (ACS) request is retried.	Integer	0 Value = 0 or empty indicates that no retry will be made.
ACS_RETRY_INTERVAL	The time (in milliseconds) to wait before another retry.	Integer	5000 Value 0 or empty indicates that there is no interval between the retries.
ACS_TIMEOUT	The time (in seconds) to wait for a response after sending an ACS request.	Integer	90
MAX_JOB	The maximum number of batch jobs.	Integer	1000
MAX_BATCH_FILE	The maximum number of batch files.	Integer	100
MAX_SESSION	The maximum number of sessions.	Integer	200



6 UDC Specific Information

6.1 UDC Specific Operations

6.1.1 Installing and Upgrading HSS Validator Plug-in

This section contains information on how to install or upgrade the HSS validator plug-in software.

Note: Make sure to have the correct plug-in available.

Also make sure that the file is named according to the syntax *<HSS Plugin Name>-<R-state>.tar.gz*

For example: *HssProvisioningValidator-R4A.tar.gz*

6.1.1.1 Installing

1. Copy the HSS Validator Plug-in software to the */home/bootloader/repository* directory on node-1.
2. Change owner and group of the HSS Plug-in file.

```
# chown actadm:activation /home/bootloader/repository/<HSS Plugin>-<R-state>.tar.gz
```

3. From node-1, run the following command for all nodes in the cluster, one by one, to add the plug-in as submodule.

```
$ bootloader.py submodule add -n <HSS Plugin Name>-<R-state>.tar.gz -t lib-ext -p dve-application --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being added.

4. From node-1, run the following command for all nodes, one by one, to activate the plug-in as submodule:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.



```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being activated.

6.1.1.2 Upgrading

1. Remove the old HSS Validator Plug-in software from the /home/bootloader/repository directory, residing on node-1.

Note: Save the file for a possible rollback, if needed later.

2. From node-1, run the following command for all nodes in the cluster, one by one, to remove the plug-in as submodule:

```
$ bootloader.py submodule delete -n <HSS Plugin Name> -p  
dve-application --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is deleted.

Note: Make sure that the file is named according to the syntax <HSS Plugin Name>

For example: HssProvisioningValidator

3. Copy the HSS Validator plug-in software to the /home/bootloader/repository directory on node-1.
4. Change owner and group of the HSS Plug-in file.

```
# chown actadm:activation /home/bootloader/repository/<H  
SS Plugin>-<R-state>.tar.gz
```

5. From node-1, run the following command for all nodes in the cluster, one by one, to add the plug-in as submodule.

```
$ bootloader.py submodule add -n <HSS Plugin  
Name>-<R-state>.tar.gz -t lib-ext -p dve-application  
--host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being added.

6. From node-1, run following command for all nodes, one by one, to activate the plug-in as submodule:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being activated.

6.1.2 Uninstalling and Rolling Back HSS Validator Plug-in

This section contains information on how to uninstall and roll back the HSS plug-in validator.

6.1.2.1 Uninstalling

1. Remove the HSS Validator plug-in software from the `/home/bootloader/repository` directory, residing on node-1.

Note: Save the file for a possible rollback, if needed later.

2. From node-1, run the following command for all nodes in the cluster, one by one, to remove the plug-in as submodule:

```
$ bootloader.py submodule delete -n <HSS Plugin Name> -p  
dve-application --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being deleted.

Note: Make sure that the file is named according to the syntax *<HSS Plugin Name>*

For example: `HssProvisioningValidator`

3. From node-1, run the following command for all nodes in the cluster, one by one, to activate the change of removing the submodule:



Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being activated.

6.1.2.2

Rolling Back

1. Remove the new HSS Validator plug-in software from the /home/bootloader/repository directory, residing on node-1.
2. From node-1, run the following command for all nodes, one by one, to remove the plug-in as submodule.

```
$ bootloader.py submodule delete -n <HSS Plugin Name> -p  
dve-application --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being deleted.

Note: Make sure that the file is named according to the syntax <HSS Plugin Name>

For example: HssProvisioningValidator

3. Copy the previously saved HSS Validator Plug-in software, see Step 1, to the /home/bootloader/repository directory, on node-1.
4. Change owner and group of the HSS Plug-in file.

```
# chown actadm:activation /home/bootloader/repository/<H  
SS Plugin>-<R-state>.tar.gz
```

5. From node-1, run the following command for all nodes, one by one, to add the plug-in as submodule.

```
$ bootloader.py submodule add -n <HSS Plugin  
Name>-<R-state>.tar.gz -t lib-ext -p dve-application  
--host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being added.



6. From node-1, run the following command for all nodes, one by one, to activate the plug-in as submodule:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node to which the submodule is being activated.

6.2 UDC Specific System Administration

6.2.1 Activating CLI

This section covers how to activate and configure CLI functions.

- For secure CLI connection (through port 8992), SSL must be configured and activated on the CLI module.
- Unsecure CLI connection is through port 8023.

6.2.1.1 CLI Module SSL Configuration and Activation

Note: Before configuring SSL for CLI module, ensure that the keystore `pgexternal_oam_keystore.jks` is created as described in Section 5.7.3.2 on page 76.

1. Set the keystore for CLI.

Edit the `/home/bootloader/config/module_config_files/UDC-Application/cli.properties` file. Uncomment the line `$keyStore=keystore_external.jks`, and edit it to: `keyStore=/home/bootloader/ssl/pgexternal_oam_keystore.jks`

2. Obtain the encrypted password.

- If the same keystore file is used as for the cudb-block-proxy module, do as follows:

```
$ bootloader.py config list -A | grep @BLOCK_PROXY
_SSLCERT_STORE_PASSWORD@
```

```
<encrypted_password>
```



- If not, obtain the password for the above keystore file and encrypt it by issuing the following command:

```
$ encrypt.sh '<clear_text_password>'
```

Note: The password needs to be inside the single quotation marks (').

Example output:

```
pwqrl5Un3PemL4v88evQ+Q==
```

3. Set keystore password for the CLI module.

Edit `/home/bootloader/config/module_config_files/UDC-Application/cli.properties` file. Uncomment the line `$keyStorePassword=` and set the encrypted password. For example:

```
keyStorePassword=pwqrl5Un3PemL4v88evQ+Q==
```

4. From node-1, activate the modifications by running the following command for each affected node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

5. Verify the certificate for SSL communication from an external node.

```
$ openssl s_client -connect <VIP-OAM-IP>:8992 -showcerts -tls1
```

Note: Below output is an example of how the `openssl s_client` can look like when a self-signed certificate negotiation is working.

Example Output:



```
CONNECTED(00000003)
depth=2 /C=US/O=VeriSign, Inc./OU=VeriSign Trust Network/OU=(c) 2006 VeriSign, Inc. - P
verify error:num=20:unable to get local issuer certificate
verify return:0
---
Certificate chain
0 s:/C=SE/ST=Stockholm/L=Stockholm/O=Ericsson/OU=IT/CN=eg8-vip-oam.ete.ka.sw.ericsson.se
i:/C=US/O=Symantec Corporation/OU=Symantec Trust Network/CN=Symantec Class 3 Secure S
-----BEGIN CERTIFICATE-----
MIIGCzCCBPogAwIBAgIQXEKFIasv3kVZjna6Ms73NjANBgkqhkiG9w0BAQsFADB+
```

6. Login as provisioning user, defined in the Dynamic Activation GUI:

```
login as: <provisioning user>
password: <password of the provisioning user>
```

CLI Prompt Output

```
>
```

6.2.1.2 Active CLI without SSL

1. Enable unsecure CLI connection through port 8023.

Edit the `/home/bootloader/config/module_config_files/UDC-Application/cli.properties` file. Modify the `allowUnsecureConnection=` line. Set value = true

Note: For secure connections, the value is set to false:

```
allowUnsecureConnection=false
```

2. From node-1, activate the modifications by running the following command for each affected node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

3. Connect to an unsecure CLI, without SSL, and login as provisioning user, defined in the Dynamic Activation GUI:

```
$ telnet <VIP-OAM-IP> 8023
```

```
login as: <provisioning user>
```



```
password: <password of the provisioning user>
```

CLI Prompt Output:

```
>
```

6.2.2 Notification Rules File Administration

The configurable notification rules files are used by Dynamic Activation to trigger notification messages towards configured notification consumers. These files contain information regarding provisioning changes of user-related data in the external database (CUDB).

Each provisioning notification rules file contains the following information:

- **Modification Information:** The modification part enables configuration of data that needs to be included when Dynamic Activation has created, changed, or deleted a subscriber profile in the CUDB.
- **Additional Information:** The additional part enables configuration of data that needs to be included in addition to those that are configured in the modification part. Typical data that is configured is subscriber identities and addresses to network entities that need to be informed about changes of the subscriber profile in the CUDB. This part is optional.
- **Send Information:** The send part enables definition of conditions that trigger the sending of the notification message to a configured notification consumer.

A check is made when a new notification is to be validated against a rule file. Once every minute, the time stamp of the file is checked. If the rule file was updated during the last minute, the rule file on disk is loaded into cache and used. The cache removes configuration rules if the files no longer exist on disk. The application server domain process does not need to be restarted.

Dynamic Activation provides notification support for IMS, EPS and DAE. The IMS and EPS files are included in the HSS-FE delivery, and the DAE file is included in the DAE-FE delivery. These files may have different names and notification purposes. It is necessary to name and store the IMS, EPS, DAE files according to the list below. For example received IMS file needs to be renamed to `NotificationRulesIms.xml`, and stored in the `/home/bootloader/config/module_config_files/NotificationRules/` directory.

The Dynamic Activation delivery includes templates for the following notification rule files:

- `/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesDae.xml`
- `/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesEps.xml`



- `/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesIms.xml`

Use the following procedure to update the notification rules:

1. Edit or replace the following notification rules configuration files for EPS, IMS, and DAE notifications, respectively:

```
/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesDae.xml
```

```
/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesEps.xml
```

```
/home/bootloader/config/module_config_files/NotificationRules/NotificationRulesIms.xml
```

Note: Folders and files can not be renamed.

2. From node-1, run the following command to deploy the customized notification rule files:

```
$ bootloader.py submodule deploy -n NotificationRules-<version>.tar.gz -t notificationrules -p dve-application --host all
```

`<version>` is the version of the `NotificationRules` module. This version can be found by running:

```
$ bootloader.py system version
```

6.2.3 Error Mapping File Administration

The error mapping is used by Dynamic Activation to map subordinate error codes found in CAI3G errors.

The configuration is performed in a file named `ErrorMappings.properties`, which is used to define the error mapping.

Dynamic Activation provides error mapping support for AVG, EPS and IMS Provisioning.

6.2.3.1 Enabling or Updating Error Mapping

To enable or update error mapping, follow the step-list:

1. Create (or if updating, modify) a file named `ErrorMappings.properties`, and add the desired error mappings into the file.

Note: If a mapping is not found in this file, the original error code and description will be used.



The mapping syntax of this file is as follows:

```
<AdditionalClassifiedError_code>#<ApplicationErrorDetails_localpart>#<Message>
= <AdditionalClassifiedError_code>
```

Parameter	Description
<AdditionalClassifiedError_code>	The error code of the additional classified error that should be mapped from.
<ApplicationErrorDetails_localpart>	The application error details localpart that should be mapped from.
<Message>	The exception message that should be mapped from. No spaces can be allowed, They should be replaced with ('.'). It is also allowed to use a wildcard ('.*'), matching any message or empty string ('')..
<AdditionalClassifiedError_code>	The new error code for the additional classified error.

The following are examples of error mappings for EPS:

```
13001#EPSFault#.* = 19001
13002#EPSFault#IMSI_defined_and_EPS_service_exists = 19002
13003#EPSFault#MSISDN_joint_to_other_IMSI.* = 19003
13003#EPSFault#MSISDN_already_provided = 19004
13003#EPSFault#IMSI_changeover_ongoing = 19005
13006#EPSFault#.* = 19006
```

For more examples, including error mappings for AVG and IMS, see the `README.txt` file, located on all nodes, in `/usr/local/pgngn/dve-application-<version>/config/errormapping`

2. Pack the `ErrorMappings.properties` file into a `.tar.gz` file:

```
$ tar zcvf ErrorMapping-<version>.tar.gz ErrorMappings
.properties
```

3. Copy the `.tar.gz` file to the `/home/bootloader/repository` directory:

```
$ sudo -u actadm cp ErrorMapping-<version>.tar.gz
/home/bootloader/repository
```

4. From node-1, run the following command to deploy the error mapping as a submodule:

```
$ bootloader.py submodule deploy -n ErrorMapping-<version>.tar.gz -t errormapping -p dve-application --host all
```

6.2.3.2 Disabling Error Mapping

To disable error mapping, after having enabled it, run the following command from node-1:



```
$ bootloader.py submodule undeploy -n ErrorMapping-<version>.tar.gz -p dve-application --host all
```

6.2.4 Configuring HTTP Max Connections

NEs using HTTP as protocol have a limit on how many connections that can be kept alive simultaneously. This can have an undesired effect of connections being opened and closed more than needed. If many southbound operations to HTTP services are used, the attribute for maximum HTTP connections, `http.maxConnections`, should be changed from the default 32 to suit the customers needs. This is changed with the following Procedure:

1. From node-1, run the following command to enable configuration of Max Connections:

```
$ bootloader.py config set --parameter @HTTP_MAX_NE_CONNECTIONS@ --value <MaxNumber>
```

2. From node-1, run the following command, for all nodes, one by one, to activate the change:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the processing node that is to be activated.

6.2.5 Configuration of MML Interface

This section is divided into two configuration parts, MML Interface configuration and MML Command configuration.

6.2.5.1 MML Interface Configuration

The MML interface configuration works the first time it gets enabled and operated, as well as possible runtime changeable parameters.

There are template files located in `/opt/dve/tools/jmxbatchclient/templates`

Use the `DVE-Inbound-MML-EJB-get.xml` template file to get the settings for the `Inbound-MML-EJB`.



Note: Output from commands is placed in `result` folder.

The first time the MML interface is deployed, the parameters in Table 28 are initiated with default values.

To be able to find and change these parameter values through Jconsole, use the following path: `com.ericsson.Configuration.DVE-Inbound-MML-EJB`

To modify these parameters using the JMX Batch Client, do the following:

1. Use the JMX Batch Client to get all current values:

```
$ sudo -u actadm /opt/dve/tools/jmxbatchclient/jbc-rmi.sh -A<provisioning client user>:<provisioning client user password> service:jmx:rmi://<PS_JMX_Server_IP>:8995/jndi/rmi://<PS_JMX_Server_IP>:8100/connector /opt/dve/tools/jmxbatchclient/templates/<XML-file with JMX data>
```

Note: When running the JMX batch client, user, and password must be specified, as first parameter with: `-A<user>:<password>`

2. Copy the output file to `DVE-Inbound-MML-EJB-set.xml` or use the template file and edit it to modify parameters:

```
$ cp <outputfile> DVE-Inbound-MML-EJB-set.xml
```

```
$ vi DVE-Inbound-MML-EJB-set.xml
```

3. Use the JMX Batch Client with `DVE-Inbound-MML-EJB-set.xml` to set the new values:

```
$ sudo -u actadm /opt/dve/tools/jmxbatchclient/jbc-rmi.sh -A<provisioning client user>:<provisioning client user password> service:jmx:rmi://<PS_JMX_Server_IP>:8995/jndi/rmi://<PS_JMX_Server_IP>:8100/connector /opt/dve/tools/jmxbatchclient/templates/DVE-Inbound-MML-EJB-set.xml
```

Note: When running the JMX batch client, user, and password must be specified, as first parameter with: `-A<user>:<password>`

4. Verify the changes by logging in to the MML interface.

Note: All runtime configuration changes on the parameters in Table 28, will take effect on the next login session. Present, already connected MML clients, are not able to see the changes.



Table 28 Runtime Configurable Parameters

Parameter	Data Type	Description
BindAddress	String	Local IP address to bind to. The <code>bindAddress</code> argument can be used on a multihomed host for a <code>ServerSocket</code> that will only accept connect requests to one of its addresses. Default value: 0.0.0.0
Echo	boolean	This command is set to <code>true</code> as default. If set to <code>false</code> , the server asks the client for using echo at startup. Default value: <code>true</code>
FirstMessage	String	The first welcome message received after a successful login procedure. Default value: MA MML PAGE 1
LineTerminationCharacter	String	Defines what the client receives for each line break. The <code>lineTerminationCharacter</code> values can be used as the following: CRNL = <code>\r\n</code> CR = <code>\r</code> NL = <code>\n</code> NONE = <code>""</code> Default value: CRNL
NotAcceptMessage	String	Fault message sent by the server to the client when the simultaneous session resources for logging in to the MML have exceeded. Default value: Max number of connections exceeded
Port	Integer	The port number listening for incoming MML traffic. Default value: 8010
Prompt	String	The prompt visible for the client after a successful login attempt. Default value: <code><</code> To add special characters in the prompt output in the XML (template) document, the following syntax must be used: <code>"</code> = <code>\03&quot</code> <code>'</code> = <code>\03&apos</code> <code><</code> = <code>\03&lt</code> <code>></code> = <code>\03&gt</code> <code>&</code> = <code>\03&amp</code>
SessionTimeout	Integer	The number of seconds a session is idle until it times out. Default value: 300



Parameter	Data Type	Description
SimultaneousSession	Integer	Maximum number of simultaneous session resources that can be connected. Default value: 160
SupressGoAhead	boolean	This command is set to <code>DO</code> as default. Inbound Telnet requests the client to use this command. Default value: <code>true</code>

Note: If the port number is changed, the Keepalived configuration, and the iptable rules need to be updated.

6.2.5.2

MML Command Configuration

The MML commands can be configured by use of the following parameters:

`ENABLE_EOT` - Certain MML commands have an 'End of Transmission' feature enabled on their response. Setting the `ENABLE_EOT` parameter to `false` disables that feature for all MML commands.

`ENABLE_CHECK_PRINTOUT` - Certain MML commands have a 'Check Printout' feature enabled on their response. Setting the `ENABLE_CHECK_PRINTOUT` parameter to `false` disables that feature for all MML commands.

`<COMMAND>_EOT` - Each command can be configured to use EOT, this is only applicable if `ENABLE_EOT` is set to `true`

`<COMMAND>_CONFIRM` - Each command can be configured to use Check Printout, this is only applicable if `ENABLE_CHECK_PRINTOUT` is set to `true`

Default configuration:



```
# This file is used to configure the behavior of MML commands
# If the command is to require a CTRL-D specify it with <command>_EOT=true,
# i.e AGAAC_EOT=true
# If the command should require a check printout, (semicolon after the command to
# execute the command) specify it with
# <command>_CONFIRM=true
#
# These settings are only valid if the global settings are set to true.

# Global setting for EOT, if this is false no command could have this behavior,
# default is true.
ENABLE_EOT=true

#Require EOT
AGAAC_EOT=true
AGSUC_EOT=true
HGAPE_EOT=true
HGLDP_EOT=true
HGMSF_EOT=true
HGPPDP_EOT=true
HGSGP_EOT=true

#Global setting for EOT, if this is false no command could have this behavior,
# default is true.
ENABLE_CHECK_PRINTOUT=true

#Requires check printout
AGAAC_CONFIRM=true
HGICC_CONFIRM=true
HGICE_CONFIRM=true
HGICI_CONFIRM=true
HGSLR_CONFIRM=true
HGIRI_CONFIRM=true
HGCSC_CONFIRM=true
HGCUC_CONFIRM=true
HGCUE_CONFIRM=true
HGCUI_CONFIRM=true
```

The `mml.properties` file is located in the `/cluster/home/bootloader/config/module_config_files/dve-inbound-interfaces-application` directory.

To configure the MML commands, follow the procedure:

1. Edit the `mml.properties` file to add specific customizations:

```
$ sudo -u actadm vi /home/bootloader/config/module_config_files/dve-inbound-interfaces-application/mml.properties
```

2. Run the following command from each node, one by one, to activate the customized `mml.properties` file:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.



```
$ bootloader.py node activate --host <hostname>
```

<hostname> is the hostname of the node that is to be activated.

6.2.6 Configuration of MML over SSH Interface

This section contains information on how the MML interface configuration works the first time it gets enabled and operated, as well as possible runtime changeable parameters.

There are template files located in `/opt/dve/tools/jmxbatchclient/templates`

Use the `DVE-Inbound-MML-EJB-SSH-get.xml` template file to get the settings for the `Inbound-MML-EJB-SSH`.

Note: Output from commands is placed in `result` folder.

The first time the MML SSH interface is deployed, the parameters in Table 29 are initiated with default values.

To be able to find and change these parameter values through Jconsole, use the following path: `com.ericsson.Configuration.DVE-Inbound-MML-EJB-SSH`

To modify these parameters using the JMX Batch Client, do the following:

1. Use the JMX Batch Client to get all current values:

```
$ /opt/dve/tools/jmxbatchclient/jbc-rmi.sh -A<provisioning client user>:<provisioning client user password> service:jmx:rmi://<PS_JMX_Server_IP>:8995/jndi/rmi://<PS_JMX_Server_IP>:8100/connector /opt/dve/tools/jmxbatchclient/templates/<XML-file with JMX data>
```

Note: When running the JMX batch client, user, and password must be specified, as first parameter with: `-A<user>:<password>`

2. Copy the output file to `DVE-Inbound-MML-EJB-SSH-set.xml` or use the template file and edit it to modify parameters:

```
$ cp <outputfile> DVE-Inbound-MML-EJB-SSH-set.xml
```

```
$ vi DVE-Inbound-MML-EJB-SSH-set.xml
```

3. Use the JMX Batch Client with `DVE-Inbound-MML-EJB-SSH-set.xml` to set the new values:

```
$ /opt/dve/tools/jmxbatchclient/jbc-rmi.sh -A<provisioning client user>:<provisioning client user password> service:jmx:rmi://<PS_JMX_Server_IP>:8995/jndi/rmi://<PS_
```



```
JMX_Server_IP>:8100/connector /opt/dve/tools/jmxbatchcli
ent/templates/DVE-Inbound-MML-EJB-SSH-set.xml
```

Note: When running the JMX batch client, user, and password must be specified, as first parameter with: `-A<user>:<password>`

4. Verify the changes by logging in to the MML interface.

Note: All runtime configuration changes on the parameters in Table 29, will take effect on the next login session. Present, already connected MML clients, are not able to see the changes.

Table 29 Runtime Configurable Parameters

Parameter	Data Type	Description
BindAddress	String	Local IP address to bind to. The <code>bindAddress</code> argument can be used on a multihomed host for a <code>ServerSocket</code> that will only accept connect requests to one of its addresses. Default value: 0.0.0.0
FirstMessage	String	The first welcome message received after a successful login procedure. Default value: MA MML PAGE 1
LineTerminationCharacter	String	Defines what the client receives for each line break. The <code>lineTerminationCharacter</code> values can be used as the following: CRNL = <code>\r\n</code> CR = <code>\r</code> NL = <code>\n</code> NONE = <code>""</code> Default value: CRNL
NotAcceptMessage	String	Fault message sent by the server to the client when the simultaneous session resources for logging in to the MML have exceeded. Default value: Max number of connections exceeded
Port	Integer	The port number listening for incoming MML traffic. Default value: 8111
Prompt	String	The prompt visible for the client after a successful login attempt. Default value: <code><</code> To add special characters in the prompt output in the XML (template) document, the following syntax must be used: " = <code>\03&quot</code> ' = <code>\03&apos</code> < = <code>\03&lt</code> > = <code>\03&gt</code> & = <code>\03&amp</code>



Parameter	Data Type	Description
SessionTimeout	Integer	The number of seconds a session is idle until it times out. Default value: 300
SimultaneousSession	Integer	Maximum number of simultaneous session resources that can be connected. Inherits the same value as for Telnet, which by default is 160.
LoginDomain	String	Defines whether DOMAIN is shown in the login prompt or not Default value: false

Note: If the port number is changed, the Keepalived configuration, and the iptable rules need to be updated.

6.2.7 Configuration of CAI Interface

This section contains information on how the CAI interface configuration works the first time it gets enabled and operated, as well as how to modify the parameters.

The first time the CAI interface is deployed, the parameters in Table 30 are initiated with default values.

To modify the default values of these parameters, do the following on node-1:

1. Run the `bootloader.py` command to list the configuration information for the system:

```
$ bootloader.py config list -A
```

Then find the CAI configuration in `nbia-module`

2. Run the `bootloader.py` command to modify the values of the parameters.

- a To set the new value, do the following:

The new value is saved in the `user_config` file.

```
$ bootloader.py config set --parameter <parameter>
--value <value>
```

Example of setting a new value for `CAI_IDLE_TIMEOUT_SERVER1`:

```
$ bootloader.py config set --parameter @CAI_IDLE_TIM
EOUT_SERVER1@ --value 200
```

```
INFO - *** @CAI_IDLE_TIMEOUT_SERVER1@=200 inserted
in user configuration
INFO - *** To activate
the changed parameter, execute "bootloader node
activate" command
```




Check `user_config` for the changed parameter:

```
$ cat /home/bootloader/config/user_config

@CAI_IDLE_TIMEOUT_SERVER1@=200
```

- b To restore the default value, do the following:

The value being restored is removed from the `user_config` file.

```
$ bootloader.py config remove --parameter <parameter>
```

Example of restoring the value for `CAI_IDLE_TIMEOUT_SERVER1`:

```
$ bootloader.py config remove --parameter
@CAI_IDLE_TIMEOUT_SERVER1@

INFO - *** Removed @CAI_IDLE_TIMEOUT_SERVER1@
from user configuration INFO - *** To activate
the changed parameter, execute "bootloader node
activate" command
```

3. From node-1, run the following command to activate the modification for each node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The `all` parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the processing node that is to be activated.

4. Verify the changes by logging in to the CAI interface.

Note:

- The new value will take effect after executed the command `bootloader node activate`
- When the CAI port number is changed, the Keepalived configuration, and the iptable rules need to be updated.

**Table 30 Configuration Parameter**

Parameter	Description	Type	Default Value
CAI_SERVER_CONNECTED_MESSAGE_SERVER1	Set of characters displayed when the connection is accepted by CAI server	String	CONNECTING TO CAI...\n\nPROCESS CaiServer1/2 CONNECTED...
CAI_SERVER_CONNECTED_MESSAGE_SERVER2			
CAI_PORT_NUMBER_SERVER1	Port number that the CAI server listens on ⁽¹⁾	Integer	3300/3301
CAI_SSL_PORT_NUMBER_SERVER2		Value range: inclusively from 1024 to 65534	
CAI_MAX_CONNECTIONS_EXCEEDED_SERVER1	Set of characters displayed when the maximum number of CAI connections exceeded	String	Max number of connections exceeded.
CAI_MAX_CONNECTIONS_EXCEEDED_SERVER2			
CAI_MAX_CONNECTIONS_SERVER1	Maximum number of CAI connections	Integer	160
CAI_MAX_CONNECTIONS_SERVER2		Value range: inclusively from 1 to 1200	
CAI_WORKER_THREAD_S1	Maximum number of CAI worker threads	Integer	32
CAI_WORKER_THREAD_S2		Value range: inclusively from 1 to 1200	
CAI_SHOW_EMPTY_ARG_SERVER1	Shows an argument in a CAI response when it does not have a value (1: to show; 0: not to show)	Enumeration value: • 0 • 1	0
CAI_SHOW_EMPTY_ARG_SERVER2			
CAI_PROMPT_SERVER1	CAI prompt displayed to the user	String	Enter command:
CAI_PROMPT_SERVER2			
CAI_REQUEST_CONFIRMATION_SERVER1	Set of characters that confirm the request has been sent	Enumeration value: • CR - uses \r • NL - uses \n • CRNL - uses \r\n • NONE	CRNL
CAI_REQUEST_CONFIRMATION_SERVER2			
CAI_MAX_LOGIN_RETRY_SERVER1	Maximum number of retries when login fails (-1 means unlimited)	Integer	-1
CAI_MAX_LOGIN_RETRY_SERVER2		Value range: inclusively from -1 to 100	
CAI_IDLE_TIMEOUT_SERVER1	Maximum time a CAI session is active when there is no interaction (seconds; 0 means infinite)	Integer (in second)	300
CAI_IDLE_TIMEOUT_SERVER2		Value range: inclusively from 0 to 3600	



Parameter	Description	Type	Default Value
CAI_CONNECTION_TY PE_SERVER1	Type of connection that CAI server uses (read only)	String	TCPIP
CAI_CONNECTION_TY PE_SERVER2			

(1) If the port number is changed, the Keepalived configuration, and the iptable rules need to be updated.

6.2.8

Configuration of EDIFACT Interface

This section contains information on how the EDIFACT interface configuration works the first time it gets enabled and operated, as well as how to modify the parameters.

The first time the EDIFACT interface is deployed, the parameters in Table 31 are initiated with default values.

To modify the default values of these parameters, do the following on node-1:

1. Run the `bootloader.py` command to list the configuration information for the system:

```
$ bootloader.py config list -A
```

Then find the EDIFACT configuration in `nbia-module`

2. Run the `bootloader.py` command to modify the values of a parameter.
 - a To set the new value, do the following:

The new value is saved in the `user_config` file.

```
$ bootloader.py config set --parameter <parameter>
--value <value>
```

Example of setting a new value for `EDIFACT_SERVER_STATUS`:

```
$ bootloader.py config set --parameter @EDIFACT_SERV
ER_STATUS@ --value disable
```

```
INFO - *** @EDIFACT_SERVER_STATUS@=disable inserted
in user configuration
INFO - *** To activate the
changed parameter, execute "bootloader.py node
activate --host <hostname>" command
```

Check `user_config` for the changed parameter:

```
$ cat /home/bootloader/config/user_config
@EDIFACT_SERVER_STATUS@=disable
```

- b To restore the default value, do the following:



The value being restored is removed from the `user_config` file.

```
$ bootloader.py config remove --parameter <parameter>
```

Example of restoring the value for `EDIFACT_SERVER_STATUS`:

```
$ bootloader.py config remove --parameter  
@EDIFACT_SERVER_STATUS@
```

```
INFO - *** Removed @EDIFACT_SERVER_STATUS@ from  
user config INFO - *** To activate the changed  
parameter, execute "bootloader.py node activate  
--host <hostname>" command duration
```

3. From node-1, run the following command to activate the modification for each node, one by one:

Attention!

Wait for each node to be activated before starting with the next one, otherwise traffic disturbances occur. The *all* parameter should only be used when no provisioning traffic is running.

```
$ bootloader.py node activate --host <hostname>
```

`<hostname>` is the hostname of the processing node that is to be activated.

4. Verify the changes by logging in to the EDIFACT interface.

Note:

- The new value will take effect after executed the command `bootloader node activate`
- When the EDIFACT port number is changed, the Keepalived configuration, and the iptable rules need to be updated.

Table 31 Configuration Parameter

Parameter	Description	Type	Default Value
EDIFACT_WORKER_THREADS	Maximum number of Dynamic Activation work threads to receiver EDIFACT requests.	Integer Value range: inclusively 0 to 100	20
EDIFACT_PORT_NUMBER	Port number that the EDIFACT server listens on. ⁽¹⁾	Integer Value range: inclusively 1024 to 65535	3010



Parameter	Description	Type	Default Value
EDIFACT_SERVER_STATUS	The status of EDIFACTE server.	String Value range: disable or enable	enable
EDIFACT_NOTIFICATIONACK_TIMEOUT	Response time-out in seconds to wait for a response from BSCS after EDIFACT sent a notification.	Integer Value range: inclusively 0 to 300	3
EDIFACT_NOTIFY_THREADS	Maximum number of notify threads that EDIFACT can process.	Integer Value range: inclusively 0 to 100	20

(1) When EDIFACT port number has been changed, new port number needs to be added to Customer Iptables Rules. For more information, see

6.2.8.1

Configuring bscs.xml for EDIFACT Interface

bscs.xml is used to configure the notification receiver on BSCS node, the configurable parameters are listed in Table 32. The execution result of EDIFACT requests is sent to the notification receivers based on the senderId in the request.

To configuration the parameters of EDIFACT receiver:

1. Edit the file /home/bootloader/config/module_config_files/nbia-module/bscs.xml. The bscs.xml file supports Multiple EDIFACT receivers.

Below is an example of configuring two notification receivers in bscs.xml:

```
<p:bscs senderId="sender01">
  <p:receiverHost>127.0.0.1</p:receiverHost>
  <p:receiverPort>60006</p:receiverPort>
  <p:usernameInNGN>example_user</p:usernameInNGN>
</p:bscs>
<p:bscs senderId="sender02">
  <p:receiverHost>127.0.0.1</p:receiverHost>
  <p:receiverPort>60007</p:receiverPort>
  <p:usernameInNGN>example_user</p:usernameInNGN>
</p:bscs>
```

2. From node-1, run the following command for all nodes in the cluster, one by one to activate.

```
$ bootloader.py node activate --host <hostname>
```

Note: The above operation may affect the provisioning.

Table 32 Configurable parameters for *bscs.xml*

Parameter	Description	Value Range
senderId	Name of the EDIFACT Receiver which is the destination EDIFACT response forwarding to.	Any effective string
receiverHost	IP address of EDIFACT notification receiver.	Inclusively 0.0.0.0 to 255.255.255.255
receiverPort	Port of EDIFACT notification receiver.	Inclusively 1024–65535
usernameInNGN	Dynamic Activation user who processes the request.	Any effective string - Dynamic Activation username with sufficient authority.

6.2.9 Load Default NE Groups and Routing Methods

This section describes how to load the default NE groups and routing methods.

1. Log on to one of the nodes.
2. Load the default NE groups and routing methods:

```
$ sudo -u actadm /usr/local/pgngn/dve-application-<version>/bin/installDefaultConfig.sh <provisioning client user>
<provisioning client user password>
```

```
Are you sure you want to load the default configuration
? This replaces existing configuration (y/n)
```

Enter **y** and press **Enter**.

6.3 UDC Data Durability

This section contains information regarding manual replay operations, and Operational Instructions if a replay fails.

6.3.1 Manual Replay Operation

This section contains information on how to run a manual replay operation using the Replay Admin Tool.

Note: Before issuing a manual replay operation, make sure that the consolidated state processing logs is in a consistent state for the indicated time span.

6.3.1.1 Running Replay Admin Tool

The commands and respective command options to use for manual replay operations are shown in Table 33.



Table 33 Manual Replay Operation Commands and Command Options

Command	Command Options	Description	Example
executeReplay	-u=<username>, --username=<username>	Username for Replay Administrator user.	replay-admin-tool.sh executeReplay -u=user -p=password -c='<replay-co mmand>' replay-admin-tool.sh executeReplay -u=user -p=password -c='<replay-co mmand>' -pb=true
	-p=<password>, --password=<password>	Password for Replay Administrator user.	
	-c=<replay cmd>, --cmd=<replay cmd>	Manual replay command to execute.	
	-pb=<true/false>, --provisioningblock= <true/false>	Set provisioning block to <true/false> until replay is complete (false by default).	
getReplaystate	-u=<username>, --username=<username>	Username for Replay Administrator user.	replay-admin-tool.sh getReplaystate -u=user -p=password
	-p=<password>, --password=<password>	Password for Replay Administrator user.	
provisioningBlock	-u=<username>, --username=<username>	Username for Replay Administrator user.	replay-admin-tool.sh provisioningBlock -u=user -p=password -a=true
	-p=<password>, --password=<password>	Password for Replay Administrator user.	
	-a=<true/false>, --active=<true/false>	Set provisioning block to <true/false> until changed (false by default).	

The replay command is as follows:

```
REPLY: STARTTIME=<POSIX time start>, ENDTIME=<POSIX time  
end>, [SOUTHBOUND=<PROTOCOL>], [TARGET=<NE-GROUP>], [ROOTLOGIDS  
="<RootLogId1, RootLogId2, ...>"];
```

Where:

- “[] ” indicates an optional parameter that can be omitted.
 - If omitted, the default value SOUTHBOUND=LDAP and TARGET=CUDB-GROUP take effect.
 - In the current release, SOUTHBOUND only supports LDAP, and TARGET must be an NE group that has LDAP connections.
- All parameters, mandatory or optional ones, can be put in any chosen order.

Note: To check if an executed replay operation is successful or failed, check for posted events or alarms for the system, see *Event and Alarm Handling*, Reference [5].

Run the Replay Admin Tool from any node:



For local use, the Replay Admin Tool executable shell script, `replay-admin-tool.sh`, is located in the `/opt/dve/bin` directory.

The following example shows how to run a manual replay operation for a specific time span:

Example 1

```
$ replay-admin-tool.sh executeReplay -u=replayuser  
-p=replaypassword -c='REPLAY:SOUTHBOUND=LDAP,STARTTIME=142  
3652362,ENDTIME=1423652555;'
```

Output:

```
Do you want to continue without setting provisioning block?  
1) Yes  
2) No, activate provisioning block for me  
3) Do not execute manual replay  
input: #? 1  
continued output:  
Manual replay accepted  
.....  
Replay is completed.
```

The following example shows how to run a manual replay operation for a specific time span, specifying a target NE Group:

Example 2

```
$ replay-admin-tool.sh executeReplay -u=replayuser  
-p=replaypassword -c='REPLAY:SOUTHBOUND=LDAP,TARGET=CUDB-G  
ROUP,STARTTIME=1423652362,ENDTIME=1423652555;'
```

Output:

```
Do you want to continue without setting provisioning block?  
1) Yes  
2) No, activate provisioning block for me  
3) Do not execute manual replay  
input: #? 1  
continued output:  
Manual replay accepted  
.....  
Replay is completed.
```

The following example shows how to run a manual replay operation for a specific time span, with selective replay using list of RootLogId(s):

Example 3

```
$ replay-admin-tool.sh executeReplay -u=replayuser -p=repl  
aypassword -c='REPLAY:SOUTHBOUND=LDAP,TARGET=CUDB-GROUP,ST  
ARTTIME=1423652362,ENDTIME=1423652555,ROOTLOGIDS="PL-31503  
301827491798CAI3G1_2,PL-31503301827491799CAI3G1_2";'
```




Output:

```
Do you want to continue without setting provisioning block?
1) Yes
2) No, activate provisioning block for me
3) Do not execute manual replay
input: #? 1
continued output:
Manual replay accepted
.....
Replay is completed.
```

The following example shows how to run a manual replay operation for a specific time span, with northbound provisioning being blocked for this command only:

Example 4

```
$ replay-admin-tool.sh executeReplay -u=replayuser
-p=replaypassword -c='REPLAY:SOUTHBOUND=LDAP,STARTTIME=142
3652362,ENDTIME=1423652555;' -pb=true
```

Output:

```
Provisioning blocked
Manual replay accepted
.....
Replay is completed.
Provisioning block removed
```

The following example shows how to query replay state for an ongoing replay operation:

Example 5

```
$ replay-admin-tool.sh getReplaystate -u=replayuser
-p=replaypassword
```

Output:

```
Replay state: REPLAY
```

6.3.2 Operational Instruction in the Event of a Failed Replay Operation

Failed replay operations render in replay alarms and events being raised, indicating the corresponding time span that is subject to manual replay operation.



Note: It is recommended to always block northbound provisioning in Dynamic Activation before using the following methodology to perform a manual replay, this is to avoid race conditions between replay operations and new service orders, affecting the same set of LDAP entries in CUDB.

In the event of replay failure, use the following steps to secure data durability for the indicated time span:

The following definitions are used in the step-list below for greater readability.

T0	Requested timestamp from when to start replaying.
T1	Timestamp when receiving replay notification from CUDB
Tnow	Current time.
T0-T1	Indicated time span.

1. Log in on node-1 or node-2.
2. Run the `replay-admin-tool.sh provisioningBlock -u=<user> -p=<password> -a=true` command, to block northbound provisioning.

For example:

```
$ replay-admin-tool.sh provisioningBlock -u=replayuser  
-p=replaypassword -a=true
```

3. Run the `replay-admin-tool.sh` command for the time span indicated in the alarm, with northbound provisioning block set.

For example:

```
$ replay-admin-tool.sh executeReplay -u=replayuser  
-p=replaypassword -c='REPLAY:SOUTHBOUND=LDAP, STARTTIME=  
1423652362, ENDTIME=1423652555; '
```

4. Run the `proclog-admin-tool.sh` command and export processing logs to CSV file format for the indicated time span (T0-T1). Convert POSIX time stamp to human readable format for the local time zone.

For example:

```
$ cd /usr/local/pgnngn/admin-tool-<version>/bin  
  
$ sudo -u actadm ./proclog-admin-tool.sh -e "2015-02-11  
10:59:22 2015-02-11 11:02:35" -p <export_dir_path>
```

5. Run the `proclog-admin-tool.sh` command and export processing logs to CSV file format ranging after the indicated time span until current time (T1-Tnow).



For example:

```
$ sudo -u actadm ./proclog-admin-tool.sh -e "2015-02-11
11:02:36 2015-02-11 12:00:00" -p <export_dir_path>
```

6. Compare the two CSV files and identify any subsequent log records using the same instance values, as for the indicated time span that is subject to replay.

If any log records were identified using the same Instance values, note the corresponding RootLogId(s) and proceed with Step 7.

7. Run the `replay-admin-tool.sh` command ranging from indicated time span until current time (T1-Tnow), with selective replay using list of RootLogId(s).

For example:

```
$ replay-admin-tool.sh executeReplay -u=replayuser
-p=replaypassword -c='REPLAY:SOUTHBOUND=LDAP,TARGET=CU
DB-GROUP,STARTTIME=1423652556,ENDTIME=1423656000,ROOTL
OGIDS="PL-31502111103491798CAI3G1_2,PL-3150211110349
1799CAI3G1_2";'
```

8. Run the `replay-admin-tool.sh provisioningBlock -u=<user> -p=<password> -a=false` command, to manually remove the block for northbound provisioning.

For example:

```
$ replay-admin-tool.sh provisioningBlock -u=replayuser
-p=replaypassword -a=false
```





7 Resource Configuration Specific Information

7.1 Use Housekeeping Script to Remove Obsolete Device Configurations

The device configurations in Cassandra database grow over time and can have a high volume. Resource Configuration provides a Housekeeping script to remove obsolete configurations safely.

7.1.1 How The Housekeeping Script Works

For a device in the Resource Configuration **Device Repository**, every the 100th configuration of the device is marked as a checkpoint in Cassandra.

When running the Housekeeping script, the script uses the current-system-time and a specified-period to find the latest checkpoint to keep, and then removes all configurations older than that checkpoint, as shown in Figure 4.

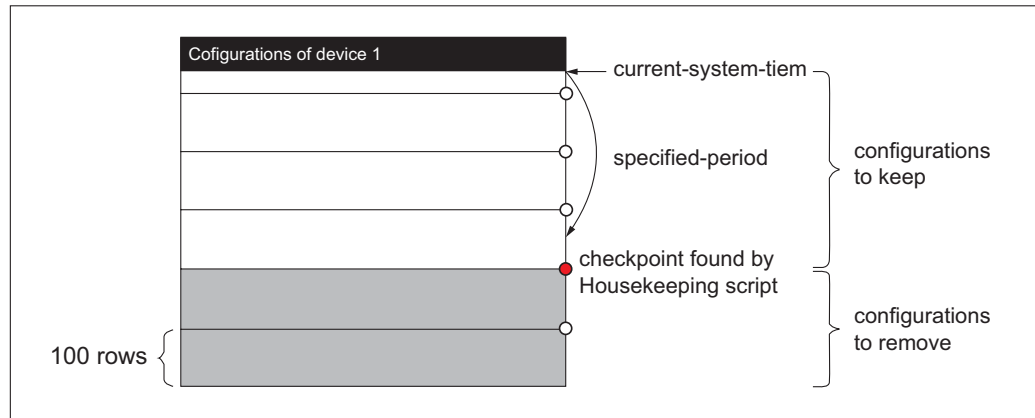


Figure 4 How the Housekeeping Script Works

Note: Because of the Cassandra configuration, the house keeping script takes effect in 10 days after execution.

The Housekeeping script removes obsolete configurations from all devices in the **Device Repository**. In the current release, users cannot choose to keep all configurations for some devices while remove obsolete configurations for others.



7.1.2 Remove Obsolete Device Configurations

Do!

Before removing obsolete device configurations, Ericsson recommends performing a backup for the Cassandra database.

For instructions, refer to *Backup and Restore Guideline for Virtual and Cloud Deployment*, Reference [10].

To remove obsolete device configuration from the **Device Repository**, do the following:

1. Log on to node-1.
2. Run the Housekeeping script, and specify a period in days, weeks, or months (greater than 0).

Device configurations within the specified-period are to be kept, and older configurations are to be removed from Cassandra. For more information, see Section 7.1.1 on page 159.

```
$ housekeeping.sh <specified period>
```

Examples:

```
$ housekeeping.sh 15 days
$ housekeeping.sh 2 weeks
$ housekeeping.sh 1 month
```

Note: The script process can be time consuming, depending on how many devices and features there are.



8 Maintenance Routines

This section describes the maintenance routines recommended to be performed on a daily, weekly, and monthly basis.

8.1 Daily Maintenance

This section holds an overview of the maintenance routines recommended to be performed daily.

- Check `/var/log/messages`.

For information about how to perform this check, see Section 4.8 on page 54.

- Check for active alarms.

For information about how to perform this check, see Section 4.8 on page 54.

- Removal of old processing logs is performed by the `proclog-retain-maximum-days.sh` crontab script.

Since the oldest logs are deleted it is recommended to export the processing logs, according to instructions in Section 5.13 on page 98.

8.2 Weekly Maintenance

This section holds an overview of the maintenance routines recommended to be performed on a weekly basis.

- Perform a system check according to instructions in Section 4.8 on page 54.
- For Resource Configuration application, check the disk utilization for `/var/cassandra/data`, see Section 4.8.1 on page 54. If the disk usage is starting to get high, run the house keeping script described in Section 7.1.2 on page 159.

8.3 Monthly Maintenance

This section holds an overview of the maintenance routines recommended to be performed on a monthly basis.

- For information about maintaining SSL certificates, see Section 4.8.3 on page 60.



- For information about backup of **Dynamic Activation Configuration**, see *Backup and Restore Guideline for Virtual and Cloud Deployment*, Reference [10].
- For information about log files, see Section 2.3.4 on page 16.



Reference List

Ericsson Documents

- [1] *Library Overview*, 18/1553-CSH 109 628 Uen
- [2] *Glossary of Terms and Acronyms*, 0033-CSH 109 628 Uen
- [3] *Function Specification Resource Activation*, 3/155 17-CSH 109 628 Uen
- [4] *CUDB Subscription Repair and Remove Procedures*, 4/1553-CSH 109 628 Uen
- [5] *Event and Alarm Handling*, 3/1553-CSH 109 628 Uen
- [6] *Network Description and Configuration for Virtual and Cloud Deployment*, 1/1551-CSH 109 628 Uen
- [7] *User Guide for Resource Activation*, 1/1553-CSH 109 628 Uen
- [8] *Configuration Manual for Resource Activation*, 2/1543-CSH 109 628 Uen
- [9] *ESA Setup and Configuration*, 1/1543-FAM 901 455 Uen
- [10] *Backup and Restore Guideline for Virtual and Cloud Deployment*, 6/1553-CSH 109 628 Uen
- [11] *Parameter List for Virtual Deployment*, 3/1057-CSH 109 628 Uen
- [12] *Parameter List for CEE Deployment*, 6/1057-CSH 109 628 Uen
- [13] *Software Installation for Virtual and Cloud Deployment*, 4/1531-CSH 109 628 Uen
- [14] *Hardening Guideline for Virtual and Cloud Deployment*, 2/154 43-CSH 109 628 Uen
- [15] *Layered HLR AUC Provisioning over MML*, 5/155 19-CSH 109 628 Uen

Online References

- [16] *The nodetool utility*, <https://docs.datastax.com/en/cassandra/2.2/cassandra/tools/toolsNodetool.html>
- [17] *OpenID Connect Core 1.0*, http://openid.net/specs/openid-connect-core-1_0.html
- [18] *OpenID Connect Session Management 1.0*, http://openid.net/specs/openid-connect-session-1_0.html