

# Core MW Management Guide

Core Middleware

USER GUIDE

**Copyright**

© Ericsson AB 2015–2018. All rights reserved. No part of this document may be reproduced in any form without the written permission of the copyright owner.

**Disclaimer**

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

**Trademark List**

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



# Contents

<b>1</b>	<b>Introduction</b>	<b>1</b>
1.1	Prerequisites	1
<b>2</b>	<b>Core MW Command Overview</b>	<b>3</b>
2.1	Deprecated Commands	6
<b>3</b>	<b>Administrative Operations</b>	<b>9</b>
3.1	Reboot Cluster	9
3.2	Reboot Node	10
3.3	Lock Node	10
3.4	Unlock Node	11
3.5	Verify System Status	11
3.6	Save IMM Data	15
3.7	Get AMF Node of Given Hostname (Core MW Classic)	15
3.8	Get Hostname of Given AMF Node	15
3.9	Configure IMM Access Control	16
3.10	Enable or Disable Core MW Features (Core MW Classic)	16
3.11	Set Time-Out for Core MW Scaling Feature	26
3.12	Clear Alarm Issued by AMF	26
3.13	Verify Instantiation Status	27
<b>4</b>	<b>Core MW Partial Backup and Partial Restore – Deprecated (Core MW Classic)</b>	<b>29</b>
4.1	Create Core MW Partial Backup – Deprecated	29
4.2	List Core MW Partial Backups – Deprecated	30
4.3	Delete Core MW Partial Backup – Deprecated	30
4.4	Restore Using Core MW Partial Backup – Deprecated	30
<b>5</b>	<b>Software Management (Core MW Classic)</b>	<b>33</b>
5.1	Import Software Bundles and Campaigns	33
5.2	Delete Software Bundles and Campaigns	34
5.3	Read from Software Inventory	34
5.4	Use SMF Campaigns	36
5.5	Configure ECIM SWM	41
5.6	Use CLI for Upgrade Package/CSP	43



5.7	Configure Reboot Behavior of Single-Step Procedures	49
<b>6</b>	<b>Configuring ISP and Remote Fencing</b>	<b>51</b>
6.1	Configure ISP Events for Cluster Restarts	51
6.2	Configure Remote Fencing Parameters	51
<b>7</b>	<b>Performance Management</b>	<b>53</b>
7.1	Create ECIM PM Job	53
7.2	Start ECIM PM Job	57
7.3	Stop ECIM PM Job	58
7.4	Delete ECIM PM Job	58
7.5	Report Status of ECIM PM Job	59
7.6	List ECIM PM Jobs	60
7.7	Modify ECIM PM Job	62
7.8	Display Active PM Instance Values	65
<b>8</b>	<b>OpenSAF Tools</b>	<b>69</b>
8.1	OpenSAF Tools Wrapper	69
<b>9</b>	<b>Backup and Restore Framework (Core MW Classic)</b>	<b>73</b>



# 1 Introduction

This document describes how to manage the Core Middleware (MW) component.

**Note:** Please note that this document describes procedures sometimes only present on Core MW Classic. For Cloud Enabled Core MW ensure that the correct module containing the commands described in the examples is also installed.

## 1.1 Prerequisites

This section describes the prerequisites, which must be fulfilled before using these procedures.

### 1.1.1 Conditions

The following conditions must apply:

- The system is ready to accept logon attempts from users.
- Commands are run as root user or prefixed with `sudo` and run by users belonging to group `system-adm`. The `sudo` configuration enforces the user to provide the password before execution of the command. Any exceptions to this are detailed in the relevant sections.





## 2 Core MW Command Overview

After successful logon the commands shown in Table 1 are available to root user or to user belonging to group system-adm and prefixed by sudo.

**Note:** The exact output from any command can differ depending on the release. Command completions can be used.

Command argument completion can be used if the OS supports it and bash shell is used.

In addition to what is listed in Table 1, all commands have a `--help` attribute.

Core MW has introduced interference checks and locking between a number of commands to minimize the risk that parallel execution disrupts the system. The following areas are included:

- Campaign execution
- Backup
- Resize of Core MW

The interference locking assures that it is only possible to execute one of these commands at the same time.

Table 1 Core MW Core Commands

Core MW Core Command	Description
<code>cmw-alarm-clear [-v] &lt;alarm-type&gt; &lt;managed-object&gt;</code>	Clear alarm issued by AMF. See Section 3.12 Clear Alarm Issued by AMF on page 26.
<code>cmw-cluster-reboot [--no-rapid-reboot] [--yes]</code>	Reboot cluster. See Section 3.1 Reboot Cluster on page 9.
<code>cmw-hostname-get &lt;amfnode&gt;</code>	Get hostname of given AMF node. See Section 3.8 Get Hostname of Given AMF Node on page 15.
<code>cmw-node-lock &lt;hostname&gt; [-t &lt;timeout&gt;   --timeout &lt;timeout&gt;]</code>	Lock node. See Section 3.3 Lock Node on page 10.
<code>cmw-node-unlock &lt;hostname&gt; [-t &lt;timeout&gt;   --timeout &lt;timeout&gt;]</code>	Unlock node. See Section 3.4 Unlock Node on page 11.
<code>cmw-node-reboot [&lt;hostname&gt;]</code>	Reboot single node. See Section 3.2 Reboot Node on page 10.



Table 1 Core MW Core Commands

Core MW Core Command	Description
<code>cmw-repository-list [--campaign] [--node] [&lt;hostname&gt;...]</code>	List imported software bundles and campaigns. See Section 5.3 Read from Software Inventory on page 34.
<code>cmw-sdp-import &lt;file&gt; [&lt;file&gt;...]</code>	Import software bundles and campaigns. See Section 5.1 Import Software Bundles and Campaigns on page 33.
<code>cmw-sdp-remove &lt;name&gt; [&lt;name&gt;...]</code>	Delete software bundles and campaigns. See Section 5.2 Delete Software Bundles and Campaigns on page 34.
<code>cmw-status [-v] &lt;class-name&gt; [&lt;class-name&gt; &gt;...]</code>	View system status. Only failing items are printed unless flag -v is specified. See Section 3.5 Verify System Status on page 11.
<code>cmw-imm-policy-set &lt;option&gt;</code>	Configure IMM access control. See Section 3.9 Configure IMM Access Control on page 16.
<code>cmw-node-alarm-timeout</code>	Set Core MW node alarm time-out. This is the time before an alarm is raised when contact with a node is lost. It is also used to determine the time when housekeeping must have finished. The housekeeping is performed after a restore when scaling is enabled (equal to the time that the system can have a faulty state after a restore).
<code>cmw-timeout-configuration</code>	Set time-out for scaling batch, node join, and parallel shutdown of Core MW scaling features. See Section 3.11 Set Time-Out for Core MW Scaling Feature on page 26.
<code>cmw-utility [-h   --help] &lt;immfind   immlist   immcfg   immadm   amfadm&gt; [&lt;options&gt;]</code>	Core MW wrapper command for OpenSAF tools. See Section 8.1 OpenSAF Tools Wrapper on page 69.

Table 2 Core MW Classic Commands

Core MW Classic Command	Description
<code>cmw-campaign-commit &lt;campaign-name&gt;</code>	Commit campaign. See Section 5.4.1 Execute an SMF Campaign on page 37.
<code>cmw-campaign-rollback &lt;campaign-name&gt;</code>	Rollback campaign. See Section 5.4.2 Roll Back a Campaign on page 39.





Table 2 Core MW Classic Commands

Core MW Classic Command	Description
<code>cmw-campaign-start [--disable-backup] &lt;campaign-name&gt;</code>	Start campaign. See Section 5.4.1 Execute an SMF Campaign on page 37. If more than one campaign is executed in sequence, <code>--disable-backup</code> can be used to inhibit backups during campaign execution, except for the first campaign where a backup is recommended.
<code>cmw-campaign-status &lt;campaign-name&gt;</code>	View campaign status. See Section 5.4.1 Execute an SMF Campaign on page 37.
<code>cmw-campaign-stop &lt;campaign-name&gt;</code>	Suspend campaign. See Section 5.4.2 Roll Back a Campaign on page 39.
<code>cmw-campaign-verify &lt;campaign-name&gt;</code>	Verify campaign. See Section 5.4.6 Initiate SMF Verification on page 41.
<code>cmw-swm</code>	Consume a UP/CSP. See Section 5.6 Use CLI for Upgrade Package/CSP on page 43.
<code>cmw-swm-config-set &lt;option&gt; [&lt;option&gt;...]</code>	Set the installation-dependent attributes in the ECIM SWM object. See Section 5.5 Configure ECIM SWM on page 41.
<code>cmw-amfnode-get &lt;hostname&gt;</code>	Get AMF node of given hostname. See Section 3.7 Get AMF Node of Given Hostname (Core MW Classic) on page 15.
<code>cmw-ait-install</code>	Consume a CSP. Used by AIT to consume a package that could not be consumed by ECIM SWM because of multiple campaigns generated based on the CBA System Model (CSM) model contained in the CSP. See Section 5.6 Use CLI for Upgrade Package/CSP on page 43.
<code>cmw-configuration &lt;option&gt;</code>	Enable or disable Core MW features. See Section 3.10 Enable or Disable Core MW Features (Core MW Classic) on page 16.

Table 3 Core MW PM Commands - These commands require the Performance Management Module

Core MW PM Command	Description
<code>cmw-pmjob-create &lt;job-name&gt; &lt;option&gt; [&lt;option&gt;...]</code>	Create an ECIM PM Job. See Section 7.1 Create ECIM PM Job on page 53.
<code>cmw-pmjob-start &lt;job-name&gt;</code>	Start an ECIM PM Job. See Section 7.2 Start ECIM PM Job on page 57.
<code>cmw-pmjob-stop &lt;job-name&gt;</code>	Stop an ECIM PM Job. See Section 7.3 Stop ECIM PM Job on page 58.



Table 3 Core MW PM Commands - These commands require the Performance Management Module

Core MW PM Command	Description
<code>cmw-pmjob-delete &lt;job-name&gt;</code>	Delete an ECIM PM Job. See Section 7.4 Delete ECIM PM Job on page 58.
<code>cmw-pmjob-modify &lt;job-name&gt; &lt;option&gt; [&lt;option&gt;...]</code>	Modify an ECIM PM Job. See Section 7.7 Modify ECIM PM Job on page 62.
<code>cmw-pmjob-status [-v] &lt;job-name&gt;</code>	View ECIM PM Job status. See Section 7.5 Report Status of ECIM PM Job on page 59.
<code>cmw-pmjob-list [&lt;option&gt;...]</code>	List all defined ECIM PM Jobs. See Section 7.6 List ECIM PM Jobs on page 60.
<code>cmw-pm-show-counters [-v] &lt;option&gt; [&lt;option&gt;...]</code>	Display active PM values for a specified instance. See Section 7.8 Display Active PM Instance Values on page 65.

## 2.1 Deprecated Commands

The deprecated commands are listed in Table 4.

Table 4 Deprecated Commands

Core MW Command	Description
<code>cmw-immSave</code>	<p>This command is deprecated and cannot be used. Using this command only result in the logging of a warning message.</p> <p>The IMM data is, starting with Core MW R2A, persistently saved automatically.</p>
<code>cmw-partial-backup-create [--verbose] &lt;label&gt;</code>	<p>This command is deprecated since Core MW R6A.</p> <p>Create Core MW partial backup. See Section 4.1 Create Core MW Partial Backup – Deprecated on page 29.</p>
<code>cmw-partial-backup-remove [--verbose] &lt;label&gt;</code>	<p>This command is deprecated since Core MW R6A.</p> <p>Delete Core MW partial backup. See Section 4.3 Delete Core MW Partial Backup – Deprecated on page 30.</p>



Table 4 Deprecated Commands

Core MW Command	Description
<code>cmw-partial-backup-restore [--verbose] &lt;label&gt;</code>	<p>This command is deprecated since Core MW R6A.</p> <p>Restore Core MW from partial backup. See Section 4.4 Restore Using Core MW Partial Backup – Deprecated on page 30.</p>
<code>cmw-partial-backup-list</code>	<p>This command is deprecated since Core MW R6A.</p> <p>List Core MW partial backups. See Section 4.2 List Core MW Partial Backups – Deprecated on page 30.</p>

Deprecated command options are listed in Table 5.

Table 5 Deprecated Command Options

Core MW Command Option	Description
<code>cmw-configuration &lt;--enable   --disable   --status&gt; ONE_STEP_UPGRADE</code>	<p>This command option is deprecated since Core MW R3A. To configure upgrade execution method, refer to Section Configure Upgrade Execution Method of the Upgrade Package in Core MW Software Management.</p> <p>Enable, disable, or check status of One Step Upgrade feature. See Section 3.10.5 One Step Upgrade Feature – Deprecated on page 20.</p>





## 3 Administrative Operations

All commands in this section are to be run as root user.

All commands except those for “Configuring IMM Access Control” can be prefixed with `sudo` and run by users belonging group `system-adm`.

This section describes the following administrative operations:

- Rebooting cluster
- Rebooting node
- Locking node
- Unlocking node
- Verifying system status
- Saving IMM data
- Getting AMF node of given hostname
- Getting hostname of given AMF node
- Configuring IMM Access Control
- Enabling or disabling Core MW features
- Setting time-out for Core MW scaling feature
- Clearing alarm issued by AMF

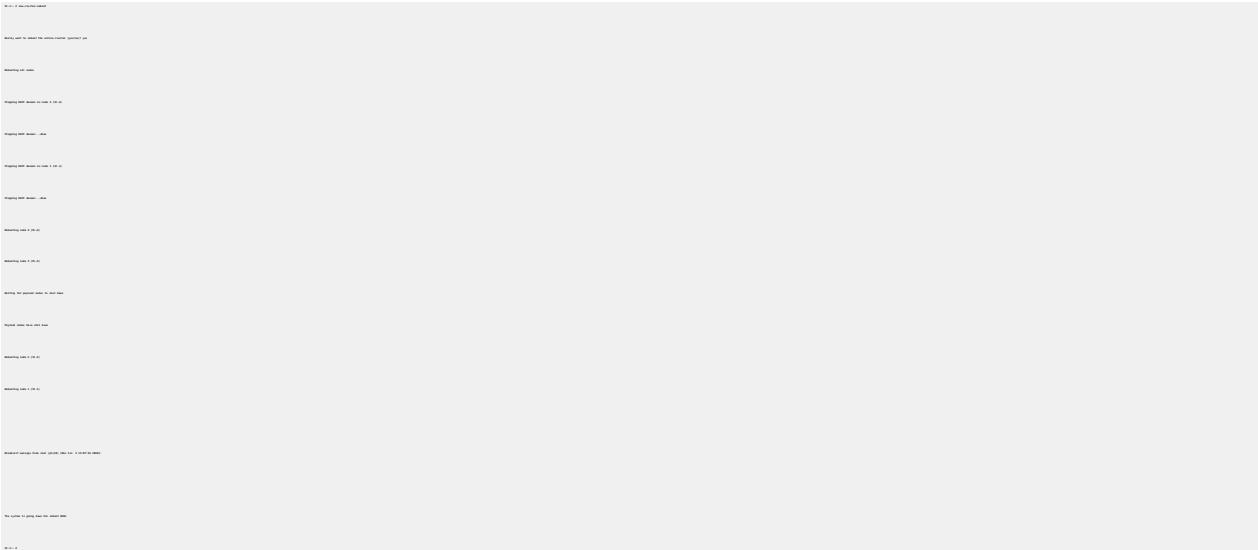
### 3.1 Reboot Cluster

To reboot the cluster in an ordered way, use the following command:

```
cmw-cluster-reboot [--no-rapid-reboot] [--yes]
```

If `--no-rapid-reboot` is specified, the cluster reboots using a regular mechanism instead of using `kexec` to facilitate fast rebooting as usual.

If `--yes` is specified, the command does not require confirmation.



Example 1 Rebooting Cluster with Interactive Confirmation

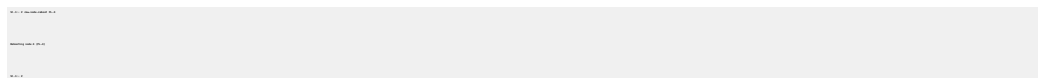
## 3.2 Reboot Node

To reboot a single node, use the following command:

```
cmw-node-reboot [<hostname>]
```

If no hostname is specified, the current node is rebooted.

**Note:** This command is normally not used, but can be necessary to recover from transient error situations on a node that cannot be resolved using procedures with less system impact.



Example 2 Rebooting Remote Node

## 3.3 Lock Node

To lock an AMF node, that is, to set its administrative state to LOCKED, use the following command:

```
cmw-node-lock <hostname> [-t <timeout> | --timeout <timeout>]
```

All Service Units (SUs) hosted on the node are prohibited from providing service.

To set the time-out in seconds, use the utility `timeout` command:



**-t, --timeout <seconds>**

If time-out is not specified, the default time-out of 900 seconds is used.

**Note:** The affected workload is redistributed according to the AMF redundancy model of the application.

Example 3 Locking Node

## 3.4 Unlock Node

To unlock an AMF node, that is, to set its administrative state to UNLOCKED, use the following command:

```
cmw-node-unlock <hostname> [-t <timeout> | --timeout <timeout>]
```

All SUs hosted on the node are administratively allowed to provide service.

To set the time-out in seconds, use the utility timeout command:

**-t, --timeout <seconds>**

If time-out is not specified, the default time-out of 900 seconds is used.

Example 4 Unlocking Node

## 3.5 Verify System Status

To verify that the system status is normal, check the status of the following system items with the following command:

```
cmw-status [-v] <class-name> [<class-name>...]
```

Here *v* is verbose and *class-name* is one of the following:

<b>app</b>	Application has an administrative state that must be verified. In a “normal” system state, the application must be in an unlocked state.
<b>csiass</b>	The Component Service Instance (CSI) has a High Availability (HA) state that must be verified. No CSI must be in quiescing or quiesced state when the system status is normal.



<b>comp</b>	<p>An AMF component has a HA state that must be verified. No component must be in quiescing or quiesced state when the system status is normal. The <code>operationalState</code> of <code>comp</code> must be <code>ENABLED</code>.</p>
<b>node</b>	<p>An executing instance of the host OS connected to the cluster. A node can be a Virtual Machine.</p>
<b>pm</b>	<p>Performance Management (PM) reports overall Ericsson Common Information Model (ECIM) PM status. If all ECIM PM Jobs are in Active state, PM returns Status OK. Otherwise PM provides details about ECIM PM Jobs that are in Stopped or Faulty state.</p> <p>The command has a short and verbose format. The short format lists only the ECIM PM Job name and state for Stopped or Faulty Jobs. The verbose format lists all ECIM PM Jobs, including those in Active states.</p>
<b>sg</b>	<p>A Service Group (SG) is a logical entity that groups one or more SUs to provide service availability for a particular set of SIs. The redundancy model defines how the SUs in the SG are used to provide service availability. Supported redundancy models are as follows:</p> <ul style="list-style-type: none"><li>• 2N – In this redundancy model one SU is active for all SIs, and one SU is standby for all SIs.</li><li>• N+M – Flexible redundancy with possibility to control the number of active and standbys. A common use of this redundancy model is the N+1 redundancy in which a single SU is standby for N active SUs.</li><li>• N-Way – In this redundancy model, it is possible for an SU to have active HA state for some SIs and at the same time have standby HA state for some other SIs.</li><li>• N-WayActive – This is a load-sharing redundancy model with only active SUs.</li><li>• No Redundancy – This redundancy model is typically used with non-critical components when the failure of a component does not cause any severe impact on the overall system and a restart is good enough.</li></ul>
<b>si</b>	<p>As components are aggregated into SUs, the AMF supports the aggregation of CSIs into a logical entity called a Service Instance (SI). An SI represents a single workload assigned to the entire SU. AMF assigns HA state to the SU on behalf of one or more of SIs.</p>





<b>siass</b>	Defines the SUs assigned to an SI. All SUs of the same type can be assigned Service Instances (SIs) derived from the same set of service types.
<b>su</b>	<p>A Service Unit (SU) is a logical entity that aggregates a set of components combining their individual functionalities to provide a higher-level service. It is the unit of redundancy in the sense that it is the smallest logical entity that can be instantiated in a redundant manner. Each SU always executes on only one node, that is, it cannot be distributed over several nodes.</p> <p>The components that constitute an SU can be developed in isolation, and a component developer can be unaware of which components constitute an SU, as they are defined at deployment time.</p>

When the system is in normal status, the output can look as shown in Example 5.

```

$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'

```

#### Example 5 System in Normal Status

When verifying system status the primary class names to specify are `su`, `node`, and `comp`. If the `si` class name is specified, `PARTIALLY_ASSIGNED` can be returned even if there is no fault in the system.

On a single node cluster (1 + 0 configuration), the output can look as shown in Example 6.

```

$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'

```

#### Example 6 Single Node System with Normal SI Status

In verbose mode on a single node cluster (1 + 0 configuration) the output can look as shown in Example 7.

```

$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'
$ su -c 'cat /etc/passwd'

```

#### Example 7 Single Node System with Normal Verbose SI Status



### Example 8 System with One Node Locked

### Example 9 Verifying System Using Verbose Mode

### Example 10 Verifying PM Job Status with One Job in State Stopped



NAME	PMJob
STATUS	Completed
START TIME	2018-06-01 10:00:00
END TIME	2018-06-01 10:05:00
PMJOB ID	PMJOB00000000000000000000000000000000
PMJOB NAME	PMJOB00000000000000000000000000000000
PMJOB TYPE	PMJOB
PMJOB STATUS	Completed
PMJOB START TIME	2018-06-01 10:00:00
PMJOB END TIME	2018-06-01 10:05:00
PMJOB ID	PMJOB00000000000000000000000000000000
PMJOB NAME	PMJOB00000000000000000000000000000000
PMJOB TYPE	PMJOB
PMJOB STATUS	Completed
PMJOB START TIME	2018-06-01 10:00:00
PMJOB END TIME	2018-06-01 10:05:00
PMJOB ID	PMJOB00000000000000000000000000000000
PMJOB NAME	PMJOB00000000000000000000000000000000
PMJOB TYPE	PMJOB
PMJOB STATUS	Completed
PMJOB START TIME	2018-06-01 10:00:00
PMJOB END TIME	2018-06-01 10:05:00

Example 11 Verifying PM Job Status Using Verbose Mode

## 3.6 Save IMM Data

The IMM data is automatically persisted incrementally for every Configuration Change Bundle (CCB) and persistent runtime object create/update/delete operation. During campaign execution, the persistency is disabled and then enabled again when the campaign reaches the completed state, that is, before commit is done.

## 3.7 Get AMF Node of Given Hostname (Core MW Classic)

To get the AMF node of a given hostname, use the following command:

```
cmw-amfnode-get <hostname>
```

NAME	AMFNode
STATUS	Completed
START TIME	2018-06-01 10:00:00
END TIME	2018-06-01 10:05:00
AMFNODE ID	AMFNODE00000000000000000000000000000000
AMFNODE NAME	AMFNODE00000000000000000000000000000000
AMFNODE TYPE	AMFNODE
AMFNODE STATUS	Completed
AMFNODE START TIME	2018-06-01 10:00:00
AMFNODE END TIME	2018-06-01 10:05:00
AMFNODE ID	AMFNODE00000000000000000000000000000000
AMFNODE NAME	AMFNODE00000000000000000000000000000000
AMFNODE TYPE	AMFNODE
AMFNODE STATUS	Completed
AMFNODE START TIME	2018-06-01 10:00:00
AMFNODE END TIME	2018-06-01 10:05:00

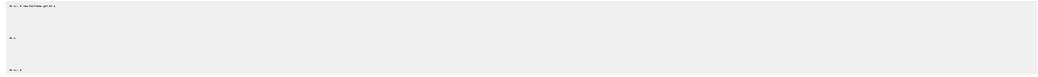
Example 12 Getting AMF Node of Given Hostname

## 3.8 Get Hostname of Given AMF Node

To get the hostname of a given AMF node, use the following command:



```
cmw-hostname-get <amfnode>
```



Example 13 Getting Hostname of Given AMF Node

## 3.9 Configure IMM Access Control

To configure IMM Access Control, use the following command:

```
cmw-imm-policy-set <enforced | disabled | permissive>
```

For compatibility reasons, the default setting of IMM Access Control is DISABLED.

By setting IMM Access Control to ENFORCED, the IMM users wanting to access IMM must belong to the `cmw-imm-users` group. For more information, refer to [Section Information Model Management Service in Core MW System Architecture Description](#).

**Note:** PERMISSIVE logs all violations to system logs without enforcing access control. Must be used with caution, as it can introduce unwanted spam in the system logs.

## 3.10 Enable or Disable Core MW Features (Core MW Classic)

To enable, disable, or check status of Core MW features, use the following command:

```
cmw-configuration <--enable | --disable | --status> <feature>
[--reboot]
```

The parameters are as follows:

- `--enable` – Enable <feature> on all nodes.
- `--disable` – Disable <feature> on all nodes.
- `--status` – Check status of <feature> (enable/disable).
- `--reboot` – Force cluster reboot immediately to enable or disable <feature> on all nodes. A cluster reboot is needed to enable or disable some features. Information about this can be found in the description of each feature.

The FENCING feature supports the following additional parameters:

- `--test` – Test specified Fencing plug-in and parameters.
- `--plugin <plug-in name>` – Specify plug-in for the FENCING feature.
- `--set <amfNodeName>` – Set `amfNodeName` for the FENCING feature.



- `--list` – List available plug-ins for the FENCING feature.

The `SHOW_COUNTERS_VERSION` feature supports the following additional parameters:

- `--select` – Select the version of the show counters API.

**Note:** `--enable` and `--disable` are not supported by the `SHOW_COUNTERS_VERSION` feature.

The following features are supported:

- `TIPC_MULTICAST`
- `SC_ABSENCE_ALLOWED`
- `SCALING`
- `ISP_REPORT`
- `ONE_STEP_UPGRADE` – Deprecated
- `PM_REPORTALWAYS`
- `ME_NAME`
- `FENCING`
- `SHOW_COUNTERS_VERSION`
- `REMOVE_DISCONNECTED_APPLIERS`

### 3.10.1

#### TIPC Multicast Feature

The Message Distribution Service (MDS) broadcasts are implemented using unicast, which adds load and does not scale on larger clusters. The Transparent Inter-Process Communication (TIPC) multicast feature solves this issue, deleting these performance impacts.

Before enabling the TIPC Multicast feature in Core MW, the user must verify that the current TIPC driver fully supports that feature.

**Note:** A cluster reboot is required for this action to take effect. An immediate cluster reboot can be triggered with parameter `--reboot`.

If the TIPC Multicast feature is already enabled, then after a Core MW upgrade it stays enabled and if it is disabled, it stays disabled.

How to enable the TIPC Multicast feature is shown in Example 14 with forced reboot, and in Example 15 without forced reboot.



#### Example 14 Enabling TIPC Multicast with Forced Reboot

#### Example 15 Enabling TIPC Multicast without Forced Reboot

If the TIPC Multicast feature is already enabled, the output is as shown in Example 16.

#### Example 16 Enabling TIPC Multicast When Already Enabled

#### Example 17 Checking TIPC Multicast Status

### 3.10.2 SC Absence Feature

The System Controller (SC) Absence feature ensures that the cluster does not reboot while both SC nodes are down, allowing the payload to work with limited service. If the time taken for the SC nodes to recover exceeds a timeout value, all payload nodes shut down (current behavior).

Before enabling the SC Absence feature in Core MW, the user must verify that the system fully supports this feature.

**Note:** A cluster reboot is required for this action to take effect. An immediate cluster reboot can be triggered with parameter `--reboot`.

`timeout` is an optional argument for the `cmw-configuration` command. By default, `timeout` has the same value as the maximum timeout (900 seconds).

If the SC Absence feature is already enabled, then after a Core MW upgrade it stays enabled and if it is disabled, it stays disabled.

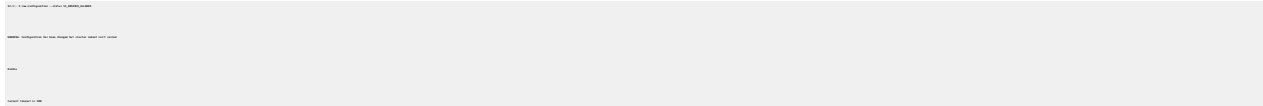
Example 18 and Example 19 are common examples to provide more information.

#### Example 18 Enabling SC Absence with Time-Out of 300 Seconds and Forced Reboot

#### Example 19 Disabling SC Absence without Forced Reboot



When the status of the SC Absence feature is checked, the timeout value is also shown. The result of checking the feature is shown in Example 20.



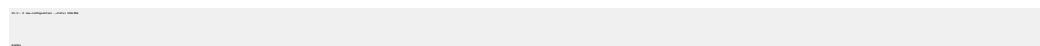
Example 20 Checking Status of SC Absence (After Change without Reboot)

### 3.10.3 Scaling Feature

When scaling is enabled, the system can automatically be expanded with newly detected nodes, using a default scaling profile. The Compute Resource Management (CRM) Northbound Interface (NBI) model allows the deletion of scalable nodes, which triggers a resize of the system. The scaling feature is enabled in runtime and does not require a reboot. When scaling feature is enabled, by default Automatic Scale-out, Manual Scale-in are enabled and Automatic Scale-in is disabled. Automatic Scale-in can only be enabled at the time of deployment and cannot be modified during runtime.

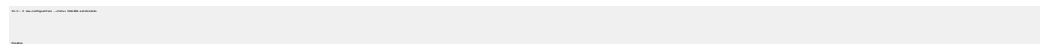
Example 21 Enabling Scaling

Example 22 shows how to check the status of the scaling feature if it is already enabled.



Example 22 Checking Status of Scaling

Example 23 Disabling Scaling



Example 24 Checking Status of Automatic Scale-In

For more details about the scaling feature, refer to [Core MW Cluster Manager Description](#).

### 3.10.4 In-Service Performance Report Feature

The ISP functionality collects ISP information that has been generated by Core MW and other components/applications (following the ISP API description, refer to ISP API Interface Description 1.1.0) and produces an XML file as output for processing by the ISP tool.



The support is as follows:

- Partial Outage reporting: partial outage events can be reported by applications or components following the specification in the ISP API. Reporting is done using the Log service.
- Network Element (NE) configuration change reporting: an upgrade/update event can be reported following the specification in the ISP API. Reporting is done using the Log service.
- ISP delivers the needed security rules to make ISP Log files visible over the NBI. These are accessed by the Automated Data Collection (ADC) tool to fetch and deliver them to the ISP tool.
- The ISP functionality keeps the XML report files for six months. Core MW automatically cleans the old ISP reports. This activity occurs once per month at the time of creation of output XML files.

Enabling/disabling the ISP report feature does not require reboot.

---

Example 25    Enabling ISP\_REPORT

---

Example 26    Checking Status of ISP\_REPORT

---

Example 27    Disabling ISP\_REPORT

**Note:** It is highly recommended to enable ISP events for cluster restarts in the system, see Section 6.1 Configure ISP Events for Cluster Restarts on page 51. Otherwise cluster restart events are not visible in the ISP logs, not even on individual blade level.

### 3.10.5    One Step Upgrade Feature – Deprecated

The Software Management Framework (SMF) normally executes the procedures in a campaign in execution level order. When the One Step Upgrade feature is enabled, the SMF collects all actions calculated to be made by the originally written procedures into a new internal single-step procedure, which executes all upgrade actions in a single step. The originally written procedures in the campaign are not executed.

---

Example 28    Enabling ONE\_STEP\_UPGRADE

Example 29 shows how to check the status of the One Step Upgrade feature if it is already enabled.






---

Example 29    Checking Status of ONE\_STEP\_UPGRADE

---



---

Example 30    Disabling ONE\_STEP\_UPGRADE

---

### 3.10.6    **SHOW COUNTERS VERSION**

The `SHOW_COUNTERS_VERSION` defines how PM instance values will be stored and reported by the PM service. When `SHOW_COUNTERS_VERSION 1` is selected, the Consumer has access to PM Job based instances. When `SHOW_COUNTERS_VERSION 2` is selected, the Consumer has access to real time PM instance data. Real time instances do not need to be associated with a PM Job.

---

Example 31    Selecting SHOW COUNTERS VERSION 1

---



---

Example 32    Selecting SHOW COUNTERS VERSION 2

---



---

Example 33    Checking Status of SHOW COUNTERS VERSION

---

### 3.10.7    **PM\_REPORTALWAYS Feature**

The `PM_REPORTALWAYS` feature allows runtime configuration of the report file content generated for active PM Measurement jobs. It affects both existing and later created PM jobs. If the feature is enabled, report content is always generated in `MISSING_MOIDS` manner, independent of the `reportContentGeneration` attribute value of each PM job. If the feature is disabled, report content is generated according to the `reportContentGeneration` attribute value of each PM job.

This feature is disabled by default.

---

Example 34    Enabling PM\_REPORTALWAYS

---



---

Example 35    Disabling PM\_REPORTALWAYS

---

Example 36 shows how to check the status of the `PM_REPORTALWAYS` feature if it is already enabled.



```
Core MW Management Guide - Core MW Management
Example 36: Checking Status of PM_REPORTALWAYS
```

Example 36 Checking Status of PM\_REPORTALWAYS

### 3.10.8 Managed Element Network Name Feature

The ME\_NAME feature allows the user to set a network name on the Managed Element (ME).

This feature is disabled by default.

```
Core MW Management Guide - Core MW Management
Example 37: Enabling ME_NAME
```

Example 37 Enabling ME\_NAME

```
Core MW Management Guide - Core MW Management
Example 38: Disabling ME_NAME
```

Example 38 Disabling ME\_NAME

Example 39 shows how to check the status of the ME\_NAME feature if it is already enabled.

```
Core MW Management Guide - Core MW Management
Example 39: Checking Status of ME_NAME
```

Example 39 Checking Status of ME\_NAME

### 3.10.9 Fencing Feature

Remote Fencing (RF) allows components, applications, and integrators to substitute the generic remote fencing logic within Core MW with a remote fencing solution more suitable to the target cluster.

Before enabling this feature, the user must verify that the system fully supports the features implemented in the RF plug-in.

To view the status of the Fencing feature, use parameter `--status` as shown in Example 40 and Example 41.

```
Core MW Management Guide - Core MW Management
Example 40: Fencing Feature Disabled
```

Example 40 Fencing Feature Disabled

```
Core MW Management Guide - Core MW Management
Example 41: Fencing Feature Enabled
```

Example 41 Fencing Feature Enabled



To view the available RF plug-ins that have been installed on the system, use parameter `--list` as shown in Example 42 and Example 43.

00.0.0.0 : 8.0.0.0 (configuration) ... status (000.000 ... 000)

### Example 42 Available RF Plug-ins Installed on System

Example 43 shows the available RF plug-ins installed on the system and that the RF plug-in is enabled but a cluster reboot is required before configuration takes effect.

```

R> plot(
+   data = data.frame(
+     x = 1:10,
+     y = 1:10,
+     z = 1:10
+   ),
+   main = "Scatter plot of x, y, and z",
+   xlab = "x",
+   ylab = "y",
+   zlab = "z",
+   xlim = c(0, 10),
+   ylim = c(0, 10),
+   zlim = c(0, 10),
+   col = "red",
+   lty = 1,
+   las = 1,
+   pch = 1,
+   cex = 1,
+   bty = "n",
+   log = "none",
+   mar = c(5, 5, 5, 5),
+   las = 1,
+   pch = 1,
+   cex = 1,
+   bty = "n",
+   log = "none",
+   mar = c(5, 5, 5, 5)
+ )

```

### Example 43 Available RF Plug-ins Installed on System – RF Plug-in Enabled

To configure and enable an RF plug-in for SC-1 and SC-2, use parameters `--enable`, `--plugin`, and `--set`, as shown in Example 44 and Example 45.

**Note:** A cluster reboot is required for this action to take effect if the RF plug-in contains any OpenSAF variable configuration. Core MW detects this automatically and informs the user that reboot is required.

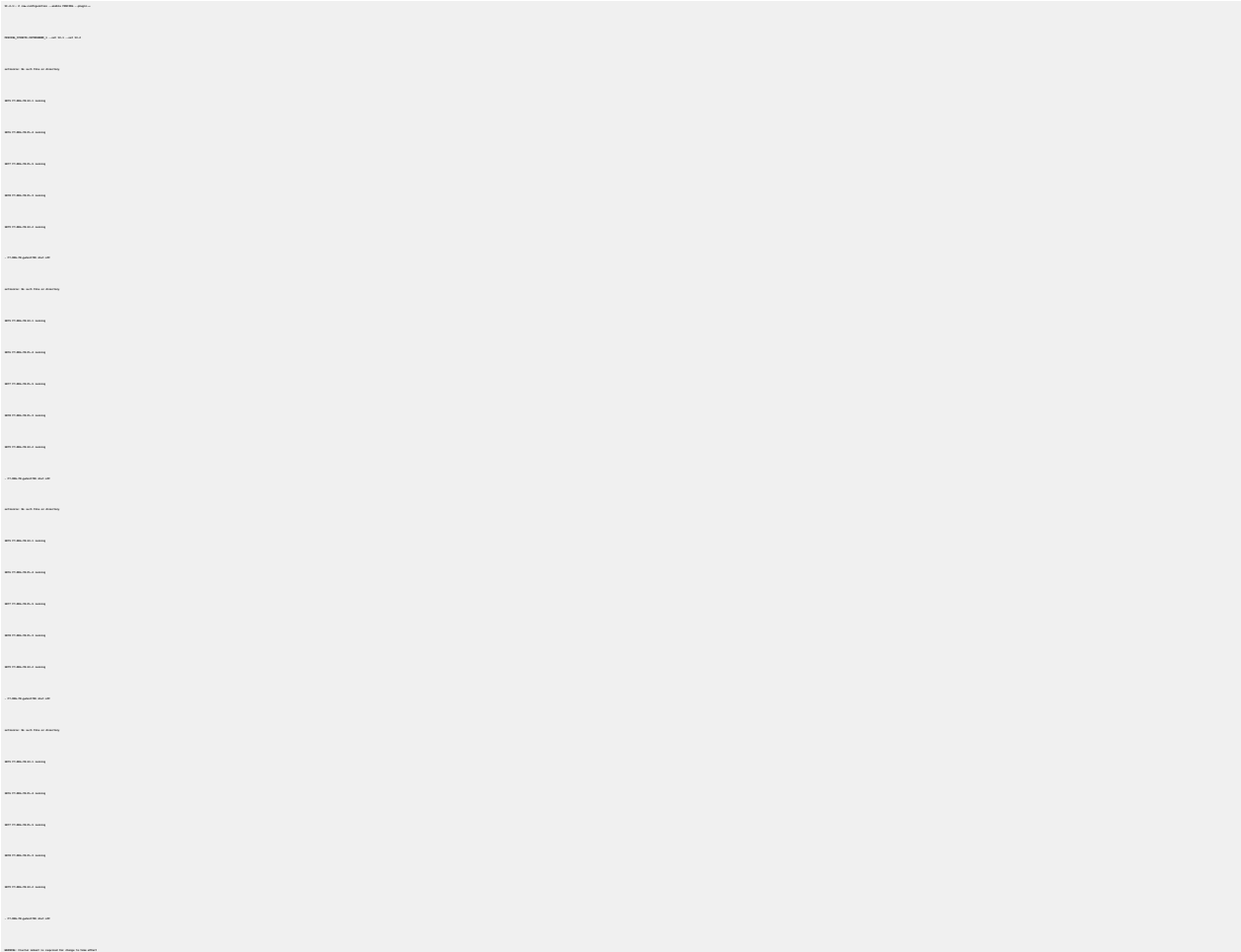
```

M.A.5: 8 the investigation --state 10000 --subset --plots --

```

### Example 44 Enabling RF Plug-in and Rebooting System

Example 45 shows how to enable an RF plug-in with the user warned that cluster reboot is required for changes to take effect.



#### Example 45 Enabling RF Plug-in and Rebooting System – Reboot Warning

Core MW performs checks on the RF plug-in before enabling it by calling the “verify” functionality of the plug-in. If any errors are detected in the RF plug-in itself, or the configured data for it, the user is alerted and further information can be found in the system logs. Example 46, Example 47, and Example 48 show some typical error use cases.



```
SC-2-1:~ # cmw-configuration --enable FENCING --reboot --plugin FENCING_IPMICXP9010010_1 =>
--set SC-1 --set SC-2
CMW: ERROR (cmw-configuration): Fencing Enable failed to verify (1)
SC-2-1:~ # less /var/log/messages
Oct 21 19:54:20 SC-2-1 FENCING_IPMI-CXP9010010_1: ERROR
(/storage/system/config/coremw/fencing/rf.FENCING_IPMI-CXP9010010_1): RFP_IPADDR,
RFP_USERID or RFP_PASSWD missing; check configuration
Oct 21 19:54:20 SC-2-1 FENCING_IPMI-CXP9010010_1: ERROR
(/storage/system/config/coremw/fencing/rf.FENCING_IPMI-CXP9010010_1): error executing
ipmitool:
Oct 21 19:54:20 SC-2-1 FENCING_IPMI-CXP9010010_1: ERROR
(/storage/system/config/coremw/fencing/rf.FENCING_IPMI-CXP9010010_1): Verify of
connection to IPMI failed
Oct 21 19:54:20 SC-2-1 CMW: Fencing plugin failed (1) on SC-1
Oct 21 19:54:20 SC-2-1 CMW: ERROR (cmw-configuration): Fencing Enable failed to
verify (1)
```

#### Example 46 Incorrect RF Plug-in Configuration Data

```
SC-2-1:~ # cmw-configuration --enable FENCING --reboot --plugin FENCING_STONITHCXP9010009_1 =>
--set SC-1 --set SC-2
CMW: ERROR (cmw-configuration): Fencing Enable failed to verify (1)
SC-2-1:~ # less /var/log/messages
Oct 21 20:02:56 SC-2-1 FENCING_STONITH-CXP9010009_1: ERROR
(/storage/system/config/coremw/fencing/rf.FENCING_STONITH-CXP9010009_1):
TARGET_PLUGIN (/cluster/home/fencingtest/target.FENCING_STONITH-CXP9010009_1-R3A20)
not readable
Oct 21 20:02:56 SC-2-1 CMW: Fencing plugin failed (1) on SC-1
Oct 21 20:02:56 SC-2-1 CMW: ERROR (cmw-configuration): Fencing Enable failed to
verify (1)
```

#### Example 47 RF Plug-in TARGET\_PLUGIN Not Found

```
SC-2-1:~ # cmw-configuration --enable FENCING --plugin FENCING_STONITH-CXP9010009_1 =>
--set SC-1 --set SC-2
CMW: ERROR (cmw-configuration): Fencing Enable failed to verify (1)
SC-2-1:~ # less /var/log/messages
Oct 21 20:07:20 SC-2-1 external/libvirt[14655]: ERROR: Failed to get status for
qemu+tcp://172.16.83.51?name=qemu:///system
Oct 21 20:07:20 SC-2-1 external/libvirt[14655]: ERROR: setlocale: No such file or
directory#012error: unexpected data 'version'
Oct 21 20:07:21 SC-2-1 stonith: external status: 'libvirt status' failed with rc 1
Oct 21 20:07:21 SC-2-1 stonith: external/libvirt device not accessible.
Oct 21 20:07:21 SC-2-1 FENCING_STONITH-CXP9010009_1: ERROR
(/storage/system/config/coremw/fencing/rf.FENCING_STONITH-CXP9010009_1): Verify of
stonith and libvirt via stonith failed, rc: 1
Oct 21 20:07:21 SC-2-1 CMW: Fencing plugin failed (1) on SC-1
Oct 21 20:07:21 SC-2-1 CMW: ERROR (cmw-configuration): Fencing Enable failed to
verify (1)
```

#### Example 48 RF Plug-in Enable Failed Because of Connection Issue to Hypervisor Management System

The user can invoke the “verify” functionality of an RF plug-in (irrespective of whether enabled or not) by entering the `cmw-configuration` command with parameter `--test`, as shown in Example 49.

```
SC-2-1:~ # cmw-configuration --test FENCING --plugin FENCING_STONITH-CXP9010009_1 --set SC-1
setlocale: No such file or directory
1075 FT-REG-70-SC-1 running
1076 FT-REG-70-PL-4 running
1077 FT-REG-70-PL-5 running
1078 FT-REG-70-PL-3 running
1079 FT-REG-70-SC-2 running
- FT-REG-70-gw01 shut off
SC-2-1:~ # echo $?
0
```

#### Example 49 Testing RF Plug-in on SC-1 – Result Is Successful



### 3.10.10 Remove Disconnected Appliers Feature

Remove disconnected appliers feature will remove an IMM applier from the system after it has been disconnected for a certain period of time.

If the feature is disabled, the legacy behaviour is to never remove disconnected IMM appliers until a cluster restart.

When the feature is enabled, the system automatically removes disconnected IMM appliers after a timeout, equal to the sum of node alarm timeout + sc absence timeout + 5 minutes.

This feature is disabled by default.

```
SC-1:~ # cmw-configuration --enable REMOVE_DISCONNECTED_APPLIERS
```

Example 50 Enabling REMOVE\_DISCONNECTED\_APPLIERS

```
SC-1:~ # cmw-configuration --disable REMOVE_DISCONNECTED_APPLIERS
```

Example 51 Disabling REMOVE\_DISCONNECTED\_APPLIERS

Example 52 shows how to check the status of the REMOVE\_DISCONNECTED\_APPLIERS feature.

```
SC-1:~ # cmw-configuration --status REMOVE_DISCONNECTED_APPLIERS
IMM Remove Disconnected Appliers: Disabled
```

Example 52 Checking Status of REMOVE\_DISCONNECTED\_APPLIERS

## 3.11 Set Time-Out for Core MW Scaling Feature

To set time-out for scaling batch, node join, and parallel shutdown of Core MW scaling features, use the following command:

```
cmw-timeout-configuration [--list | --set <timeout> <value>]
```

The parameters are as follows:

- --list – View the value of these time-outs.
- --set – Set a value to these time-outs through arguments SCALING\_BATCH, SCALING\_JOIN, and SCALING\_SHUTDOWN.

Example 53 Setting Time-Out Value for Node Join Cluster

## 3.12 Clear Alarm Issued by AMF

To clear an alarm issued by AMF, use the following command:

```
cmw-alarm-clear [-v] <alarm-type> <managed-object>
```



Here *v* is verbose, *managed-object* is the Distinguished Name (DN) of the alarming object, and *alarm-type* is one of the following:

<b>cleanup</b>	To clear an alarm of type “AMF Component Cleanup Failed” that was raised when AMF did not successfully clean up a software component.
<b>instantiation</b>	To clear an alarm of type “AMF Component Instantiation Failed” that was raised when AMF did not successfully instantiate a software component.
<b>model_error</b>	To clear an alarm of type “MDF Detected Model Error” that was raised when Core MW MDF detected an error during model delivery.
<b>proxy</b>	To clear an alarm of type “Proxy Status of a Component Changed to Unproxied” that was raised when a component, that was previously being proxied, had currently no proxy component mediating for it.
<b>unassign</b>	To clear an alarm of type “AMF SI Unassigned” that was raised when an SI had no active assignments to any SU.

#### Example 54 Clearing Alarm

### 3.13 Verify Instantiation Status

To monitor offline image creation and instantiation status, use the following command:

```
cmw-status instantiation
```

Table 6 shows the return status when the command is executed:

Returned status	Description
BOOTSTRAP_INSTANTIATION	Preparing for bootstrap
GENERATING_CONFIGURATION	Create node configuration
WAITING_FOR_CLUSTER	Waiting for cluster up
PREPARING_STARTUP	Generating campaign for start up
STARTING_SYSTEM	Executing campaign
Status OK	Instantiation successful
INSTANTIATION_FAILED	There is an error during the execution



```
SC-1:~ # cmw-status instantiation
Status OK
```

Example 55 Check instantiation status in successful case





## 4 Core MW Partial Backup and Partial Restore – Deprecated (Core MW Classic)

**Note:** This section and all its subsections contain information that is deprecated since Core MW R6A.

When the scaling feature is enabled, all the following mentioned Core MW partial backup commands are disabled.

This section describes the following partial backup and partial restore operations:

- Creating Core MW partial backup
- Listing Core MW partial backups
- Deleting Core MW partial backup
- Restoring using Core MW partial backup

A Core MW partial backup contains a snapshot of files for Core MW, which represents the system state of Core MW. The scope of a Core MW partial backup can be extended if application programs register application-specific backup commands. The partial backup mainly acts as a driver of backup where the OS and the applications keep their backup files on their own.

The Core MW partial backup can usually be used to restore a system, but as it does not contain all files it cannot be used for restoring the system after a catastrophic event.

Core MW supports integration of custom backup manager frameworks. If a custom backup manager is registered, the responsibility for backup and restore is transferred to the backup manager. The **cmw-partial-backup-create** and **cmw-partial-backup-register** commands are disabled and returns non-zero error codes when executed.

### 4.1 Create Core MW Partial Backup – Deprecated

To create a Core MW partial backup, use the following command:

```
cmw-partial-backup-create [--verbose] <label>
```

A partial backup with the specified label is created.

The label can be a maximum of 128 characters and can only contain characters that are valid in a Linux® filename.

The partial backup is physically divided in multiple archives files. The partial backup covers the following areas:



- Host OS
- Core MW
- Software bundles and campaigns imported to Core MW repository
- Data provided by registered application backup commands

```
SC-1:~ # cmw-partial-backup-create mgmtug-example  
Snapshot starting (/cluster/snapshot/.cmwea-snapshot-mgmtug-example.tar.gz)  
Snapshot completed  
SC-1:~ #
```

Example 56 Creating Core MW Partial Backup

## 4.2 List Core MW Partial Backups – Deprecated

To list the labels of the created partial backups, use the following command:

**cmw-partial-backup-list**

Only the labels of the complete partial backups are listed. For incomplete labels of partial backups, a warning message is printed.

```
SC-1:~ # cmw-partial-backup-list  
cmw-partial-backup-list: warning: no partial backup labels found  
SC-1:~ #
```

Example 57 Listing Core MW Partial Backup Labels

## 4.3 Delete Core MW Partial Backup – Deprecated

To delete a previously created partial backup, use the following command:

**cmw-partial-backup-remove [--verbose] <label>**

```
SC-1:~ # cmw-partial-backup-remove <label>  
SC-1:~ #
```

Example 58 Deleting Core MW Partial Backup

## 4.4 Restore Using Core MW Partial Backup – Deprecated

To restore the system from a previously created partial backup, use the following commands:

**cmw-partial-backup-restore [--verbose] <label>**



```
cmw-cluster-reboot --yes
```

The partial backup must be activated by rebooting the cluster, see Section 3.1 Reboot Cluster on page 9.

```
SC-1 # cmw-partial-backup-restore mgmtug-example
Restore the Host OS ...
Snapshot restore starting (/cluster/snapshot/cmwea-snapshot-mgmtug-example.tar.gz)
Snapshot restore completed
Starting RPM synchronization of node 1
Synchronizing linux-control-R1A03-PRE1.x86_64.rpm
Synchronizing =>
COREMW_COMMON-R1A-72.x86_64.716a0b126ae0b34d2f841e5cd3c539e3.rpm
Synchronizing =>
coremw-opensaf-4.0-R1A01.x86_64.09ade38a707f803470fdcf58d92f5434.rpm
Synchronizing =>
opensaf-amf-libs-4.0.RC1-R1A01.1580.4.x86_64.a8774057970069565b178c62002b893f.rpm
Synchronizing =>
opensaf-amf-nodedirector-4.0.RC1-R1A01.1580.4.x86_64.a772b062ed3638ac48a59e8bfd30e2db.rpm
Synchronizing =>
opensaf-ckpt-libs-4.0.RC1-R1A01.1580.4.x86_64.b40335b7cfa2f3592f08246fad13844c.rpm
Synchronizing =>
opensaf-ckpt-nodedirector-4.0.RC1-R1A01.1580.4.x86_64.14866c4e3f964cda83e97d673b7447.rpm
Synchronizing =>
opensaf-clm-libs-4.0.RC1-R1A01.1580.4.x86_64.1574c1c5e29562731573eb045d45d520.rpm
Synchronizing =>
opensaf-clm-nodeagent-4.0.RC1-R1A01.1580.4.x86_64.b7c8544f5dd86d7d33e5b8575ae3fe00.rpm
Synchronizing =>
opensaf-imm-libs-4.0.RC1-R1A01.1580.4.x86_64.59f8198c1ccbcd5ef6f21e464e19b4a9.rpm
Synchronizing =>
opensaf-imm-nodedirector-4.0.RC1-R1A01.1580.4.x86_64.769e124e417a0711c4dd4771a48a3c26.rpm
Synchronizing =>
opensaf-libs-4.0.RC1-R1A01.1580.4.x86_64.e23173b8021f50927f1328a299f793f4.rpm
Synchronizing =>
opensaf-log-libs-4.0.RC1-R1A01.1580.4.x86_64.2fbc9be1867d7c307351aeb5ebba9425.rpm
Synchronizing =>
opensaf-ntf-libs-4.0.RC1-R1A01.1580.4.x86_64.ded91cefd458156125bed888782afe73.rpm
Synchronizing =>
opensaf-smf-libs-4.0.RC1-R1A01.1580.4.x86_64.11a6b0cad1300999bf9dfa8ce4bef3d6.rpm
Synchronizing =>
opensaf-smf-nodedirector-4.0.RC1-R1A01.1580.4.x86_64.cf8cce9975734cf5d0324a24c1565ff1.rpm
Synchronizing =>
opensaf-4.0.RC1-R1A01.1580.4.x86_64.77e4dc0c1c15c20cac63097935f54cfd.rpm
Synchronizing =>
opensaf-amf-director-4.0.RC1-R1A01.1580.4.x86_64.a79972ce3e15bf7c307526459e00349f.rpm
Synchronizing =>
opensaf-ckpt-director-4.0.RC1-R1A01.1580.4.x86_64.6f6c5eaf5f6b413b4c4b88fbb65ef9b4.rpm
Synchronizing =>
opensaf-clm-server-4.0.RC1-R1A01.1580.4.x86_64.fa5baf4a1fa773e50037bb7afc5c679f.rpm
Synchronizing =>
opensaf-controller-4.0.RC1-R1A01.1580.4.x86_64.ba49b887419f2c4753f4fda58024f758.rpm
Synchronizing =>
opensaf-imm-director-4.0.RC1-R1A01.1580.4.x86_64.41b0fb79df6fd3fa25076aa43b291fce.rpm
Synchronizing =>
opensaf-log-server-4.0.RC1-R1A01.1580.4.x86_64.308b7372726cb676e9e51da7d4dbca45.rpm
Synchronizing =>
opensaf-ntf-server-4.0.RC1-R1A01.1580.4.x86_64.932bfbc668a973b6941054e99416988f.rpm
Synchronizing =>
opensaf-smf-director-4.0.RC1-R1A01.1580.4.x86_64.50ec99d3b2b0a6a3fafdbec9257d5b43.rpm
Synchronizing =>
COREMW_SC-R1A-53.x86_64.679a7116d9d5a9758c0c8f081c99ed5b.rpm
Completed RPM synchronization of node 1
.....
Completed RPM synchronization of node 9
Unpack the Core MW backup ...
Restore application data [backup-ERIC-Vip-CXP9013048_4-R1A03] ...
SC-1 # cmw-cluster-reboot --yes
```

## Example 59 Restoring Using Core MW Partial Backup





## 5 Software Management (Core MW Classic)

This section describes the following software management tasks:

- Importing software bundles and campaigns
- Deleting software bundles and campaigns
- Reading from software inventory
- Using SMF campaigns
- Configuring ECIM SWM
- Using CLI for upgrade package/CSP
- Configuring ISP events for cluster restarts
- Configuring reboot behavior of single-step procedures
- Configuring remote fencing parameters

Commands are to be run as root user or prefixed with `sudo` and run by a user belonging to the `system-adm` group.

### 5.1 Import Software Bundles and Campaigns

To import software bundles or campaigns to the repository, use the following command:

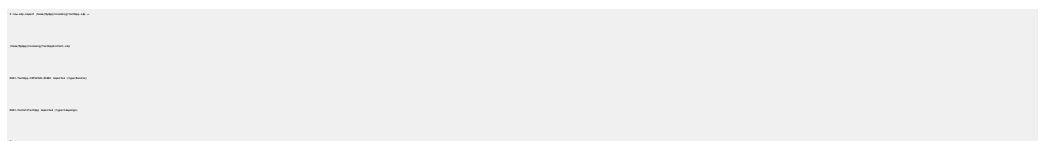
```
cmw-sdp-import <file> [<file>...]
```

The possible formats are the following:

- Bundle Software Delivery Package (SDP)
- Campaign SDP
- Bundle RPM

**Note:** Different file formats can be mixed in the command.

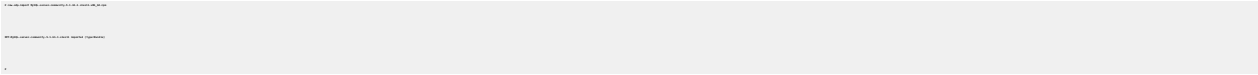
The names of successfully imported software bundles and campaigns are printed.



Example 60 Importing Bundle SDP



A third party product (3PP) software import in bundle RPM format is shown in Example 61.



Example 61 Importing Bundle RPM

## 5.2 Delete Software Bundles and Campaigns

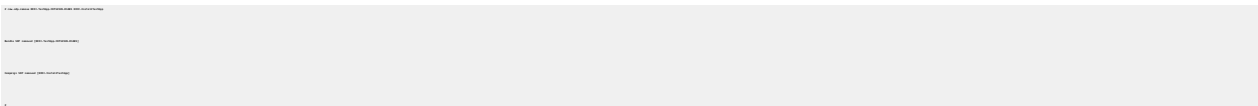
To delete software bundles and campaigns from the repository, use the following command:

```
cmw-sdp-remove <name> [<name>...]
```

The possible formats are the following:

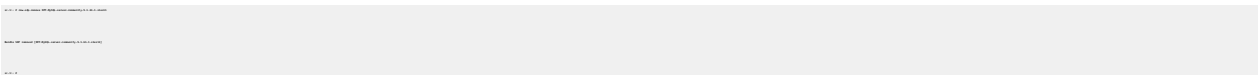
- Bundle SDP
- Campaign SDP
- Bundle RPM

**Note:** Different formats can be mixed in the command.



Example 62 Deleting Bundle SDP

**Note:** Deleting already deleted software bundles or campaign SDPs is not considered an error.



Example 63 Deleting Bundle RPM

## 5.3 Read from Software Inventory

To list the imported campaigns, use the following command:

```
cmw-repository-list --campaign
```



```

# Example 64: Listing Imported Campaigns

# Command: cmw-repository-list

# Output:

```

#### Example 64 Listing Imported Campaigns

To list the imported software bundles and if they are used or not in the system, use the following command:

**cmw-repository-list**

A list of imported software bundles and if they are used or not in the system is shown in Example 65.

```

# Example 65: Reading from Software Inventory

# Command: cmw-repository-list

# Output:

```

#### Example 65 Reading from Software Inventory

The first column shows the name of the software bundle and the second column states whether the software bundle is used, without specifying the node.

To list all software bundles installed per node, use the following command:

**cmw-repository-list --node [<hostname>...]**

If hostname is not used, the output looks as in Example 66.

```

# Example 66: Reading from Software Inventory Using Variable Node

# Command: cmw-repository-list --node <hostname>...

# Output:

```

#### Example 66 Reading from Software Inventory Using Variable Node



## 5.4 Use SMF Campaigns

All kinds of software reconfiguration such as installation, upgrade, and deletion are done with a campaign.

A campaign is contained and delivered in an SDP. To make the campaign available to Core MW, the campaign SDP is imported using the `cmw-sdp-import` command. For more information about this command, see Section 5.1 Import Software Bundles and Campaigns on page 33.

To list imported campaigns, use the following command:

```
cmw-repository-list --campaign
```

For more information about the command, see Section 5.3 Read from Software Inventory on page 34.

The software reconfiguration specified in a `campaign.xml` file is executed by SMF. SMF implements a state machine that interprets the XML file and executes the software reconfiguration.

The transitions between SMF states depending on Core MW commands are shown in Figure 1.



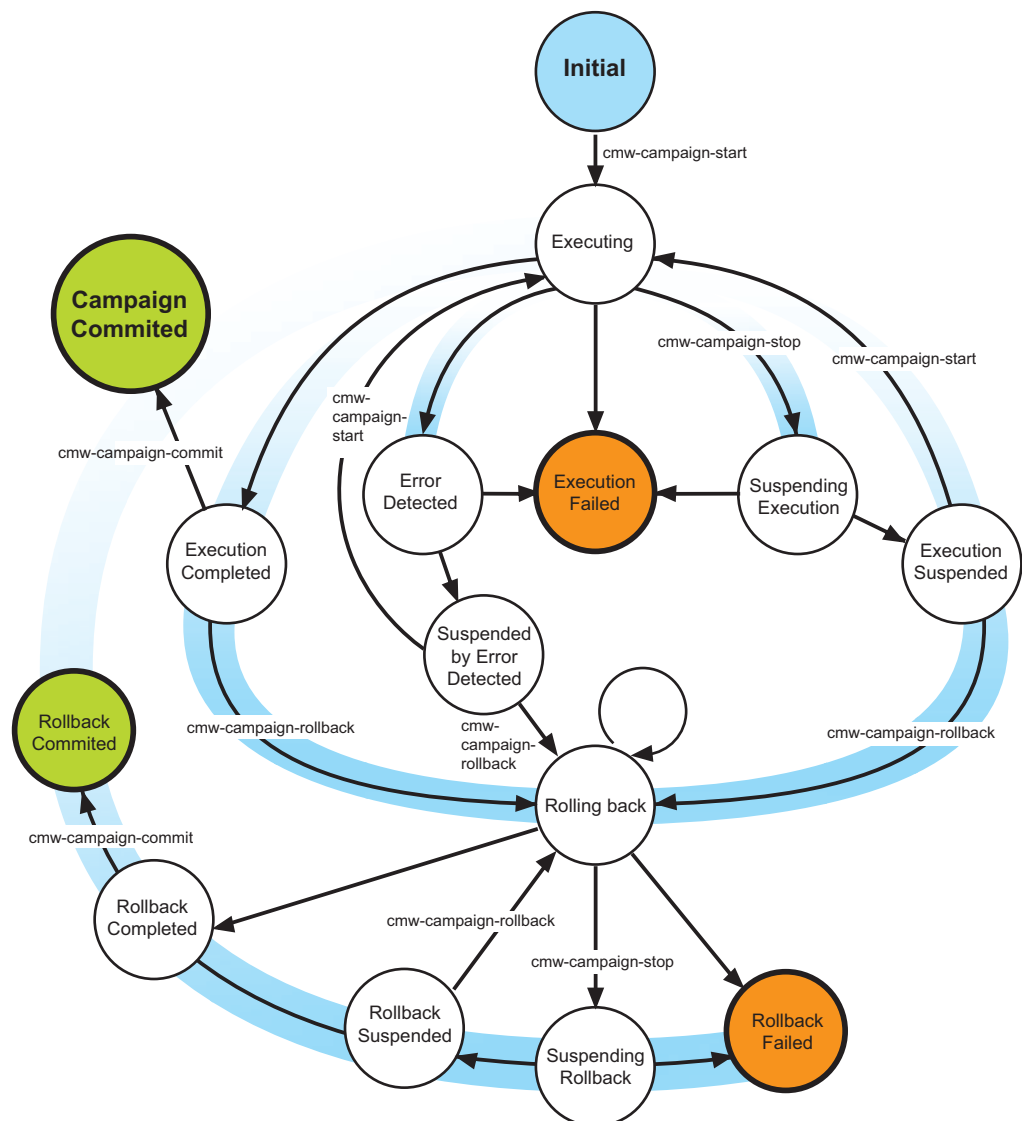


Figure 1 SMF State Diagram

### 5.4.1 Execute an SMF Campaign

Ensure that the units (node, SU, component, and so on) are affected by the campaign and that they are free of faults before executing.

To execute a campaign:

1. Get a list of the installed campaigns:

```
cmw-repository-list --campaign
```

2. Start an SMF campaign:

```
cmw-campaign-start <campaign-name>
```



If the SMF campaign is a template, the first step performed is that the campaign is updated with information fetched from the system. Any error that occurs during campaign update and validation results in an error message and the campaign execution is not started. If this occurs, a new campaign SDP must be provided, but no other measures are needed.

If more than one campaign is executed in sequence, the optional parameter `--disable-backup` can be used to inhibit backups during campaign execution. However, when the first campaign is started, a backup is recommended and the optional parameter must not be used.

**Note:** Execution of a new campaign cannot be started until the previous campaign is committed.

3. Verify that the campaign status has state COMPLETED:

```
cmw-campaign-status <campaign-name>
```

If SMF detects that a campaign cannot be initiated, the initial transition is `Cannot_initiate` and the state remains `Initial`. The `Cannot_initiate` transition can be caused by an invalid campaign file, or a campaign that executes already. Once executing, the campaign can either be successful or it can fail.

**Note:** As one of the first steps of campaign execution a backup is taken. A failed campaign requires that the system is restored from a backup, during which there is a service outage.

The backup is labeled according to the following syntax:

```
SMF-BACKUP_YYYY-MM-DDTHH:MM:SS
```

Examples:

```
SMF-BACKUP_2010-07-05T12:48:08  
SMF-BACKUP_2010-07-05T12:58:56
```

Depending on the state, take different actions, as follows:

State	Action
INITIAL without error	Retry status command.
INITIAL with error	Contact the campaign provider.
EXECUTING	Retry status command.
ERROR_DETECTED	Retry status command.
SUSPENDED_BY_ERROR_DETECTED	Start or rollback.



State	Action
COMPLETED	Proceed with commit step.
FAILED	Initiate a fallback by restoring and restarting the cluster, see Section 4.4 Restore Using Core MW Partial Backup – Deprecated on page 30.

4. Verify the status of the cluster, see Section 3.5 Verify System Status on page 11.

If the status is OK, continue with the next step, otherwise repeat the status check.

5. Commit the campaign, if it executed successfully:

```
cmw-campaign-commit <campaign-name>
```

6. Verify that the campaign status is COMMITTED:

```
cmw-campaign-status <campaign-name>
```

It can take some time before the campaign is committed. Repeat the command until the campaign status is COMMITTED.

## 5.4.2 Roll Back a Campaign

To roll back a campaign:

1. Stop the executing campaign, if the campaign is executing:

```
cmw-campaign-stop <campaign-name>
```

2. Roll back the campaign:

```
cmw-campaign-rollback <campaign-name>
```

3. Verify that the campaign status is ROLLBACK COMPLETED:

```
cmw-campaign-status <campaign-name>
```

If the campaign status is ROLLBACK COMPLETED, continue with the next step.

If the campaign status is ROLLBACK FAILED, initiate a fallback by restoring and restarting the cluster, see Section 4.4 Restore Using Core MW Partial Backup – Deprecated on page 30.

4. Commit the campaign, if it rolled back successfully:

```
cmw-campaign-commit <campaign-name>
```

5. Verify that the campaign status is ROLLBACK\_COMMITTED:



```
cmw-campaign-status <campaign-name>
```

It can take some time before the campaign is committed. Repeat the command until the campaign status is ROLLBACK\_COMMITTED.

### 5.4.3 Add Software

Applications are installed using a campaign. The application is contained in one or more software bundles and the installation campaign in an SDP. The installation campaign is target system size specific.

To install an application using a campaign:

1. Copy the installation campaign SDP and the software bundles to a location at the target system, for example:

```
/home/MyApp1/incoming
```

2. Import all campaign SDPs and software bundles:

```
cmw-sdp-import /home/MyApp/incoming/<installation-campaign-filename>
```

```
cmw-sdp-import /home/MyApp/incoming/<application-filename>
```

3. Execute the installation campaign, see Section 5.4.1 Execute an SMF Campaign on page 37.
4. Delete the installation campaign SDP:

```
cmw-sdp-remove <installation-campaign>
```

### 5.4.4 Upgrade Software

Any software delivered as a software bundle is upgraded using a campaign. The new version of the software is contained in a software bundle and the upgrade campaign in a campaign SDP.

To upgrade software using a campaign:

1. Copy the campaign SDP and the software bundles to a location at the target system, for example:

```
/home/MyApp1/incoming
```

2. Import all campaign SDPs and software bundles:

```
cmw-sdp-import /home/MyApp/incoming/<upgrade-campaign-filename>
```

```
cmw-sdp-import /home/MyApp/incoming/<new-software-filename>
```

3. Execute the upgrade software campaign, see Section 5.4.1 Execute an SMF Campaign on page 37.



4. Delete the campaign SDP and the old application software bundles:

```
cmw-sdp-remove <upgrade-campaign> <old-software-bundle-name>...
```

### 5.4.5 Delete Software

Software is deleted using a campaign. The removal campaign is contained in an SDP and target system specific.

To delete software using a campaign:

1. Import the removal campaign:

```
cmw-sdp-import /home/MyApp/incoming/<removal-campaign-sdp-filename>
```

2. Execute the removal campaign, see Section 5.4.1 Execute an SMF Campaign on page 37.
3. Delete the removal campaign SDP and the software bundles:

```
cmw-sdp-remove <removal-campaign> <software-bundle-name>...
```

### 5.4.6 Initiate SMF Verification

An application can provision itself with the ability to influence the progress of an SMF campaign by registering a callback function during application startup. Once the callback is registered, it is called whenever a campaign is initiated, see Section 5.4.1 Execute an SMF Campaign on page 37.

To initiate campaign verification manually and hence execute all currently registered callbacks, use the following command:

```
cmw-campaign-verify <campaign-name>
```

The command returns exit code 0 on success, otherwise exit code 1.

Verification can also be initiated through ECIM SWM `verify()`, which makes use of `cmw-campaign-verify`.

## 5.5 Configure ECIM SWM

If ECIM for SWM is to be used, the installation-dependent attributes in the ECIM SWM object must be set. These attributes are described in Table 7.

For more information about these attributes, refer to [Core MW Software Management and Managed Object Model cmw\\_SwM](#).

All these attributes can be set using the following command:

```
cmw-swm-config-set <option> [<option>...]
```



Except for `localFileStorePath` and `subType`, all attributes in Table 7 can also be configured using the NBI.

The command takes the options described in Table 7.

Table 7 Command Options

Option	Description
<code>-l, --localFileStorePath</code>	Deprecated option <code>localFileStorePath</code> is not used.
<code>-t, --tmpStorePath</code>	<p><code>tmpStorePath</code> is the path to the temporary storage to be used by ECIM SWM during the upgrade process. If <code>tmpStorePath</code> is not set, ECIM SWM uses the <code>/tmp</code> directory as the default temporary storage.</p> <p>Expectations on the temporary storage are as follows:</p> <ul style="list-style-type: none"><li>• The directory structure must be created before using <code>cmw-swm-config-set</code>.</li><li>• The directory must be writable to ECIM SWM.</li></ul>
<code>-s, --subType</code>	<p><code>subType</code> is the name of the node type. The <code>subType</code> describes the Managed Element, for example, "smallSystem" or "mergedSystem". This attribute is used to select the correct campaign to apply from a UP.</p> <p>For more details about how it works, refer to Section Element MESubtype in <i>Core MW Software Management</i>.</p>
<code>-b, --automaticBackup</code>	<p>Set <code>automaticBackup</code> to 1 (true) if ECIM SWM is to take an automatic backup at activation, otherwise use 0 (false).</p> <p>Defaults to 1 (automatic restore enabled).</p>
<code>-r, --automaticRestore</code>	<p><code>automaticRestore</code> is set to 1 (true) in order for ECIM SWM to perform automatic restore at failure.</p>
<code>-f, --fallbackTimer</code>	<p><code>fallbackTimer</code> specifies the maximum number of seconds ECIM SWM waits for user action before fallback. Defaults to 1200.</p>
<code>-a, --alarmBeforeTimeout</code>	<p><code>alarmBeforeTimeout</code> specifies the time in seconds that an alarm is issued before a fallback occurs. Defaults to 180.</p>

#### Example 67 Configuring ECIM SWM

The command activates automatic backup, sets `subType` to "smallCluster", and the local file store path to `/home/my/package/repository`.



## 5.6 Use CLI for Upgrade Package/CSP

Core MW can consume a UP/CBA Software Package (CSP) by using the command `cmw-swm`. This command allows users not using the COM component to create a UP/CSP and upgrade a CBA system using the produced package. It also allows components and applications to use the same upgrade scenario with a limited CBA stack deployed. The command takes the options described in Table 8.

Table 8 Command Options

Option	Description	Example
<code>--up-list</code>	View a list ID of UPs or CSPs in the system. The number following the UP ID is the action ID of the Create action that created the UP.	<pre>\$ cmw-swm --up-list</pre> Output: ERIC-Test-UP150922123955-P1A01 1



Table 8 Command Options

Option	Description	Example
--up-create	<p>Create a new UP/CSP instance from URI – points to the directory where the UP/CSP content is located. The URI is local (for example file://data/dir/subdir, /storage/system/.../UP.tgz) or a remote (for example sftp://hostname/dir/subdir).</p> <p>A UP is active when it has been created but not deleted or confirmed. No other UPs can be created if another UP is active. If the user tries to do that, error messages similar to the following are displayed:</p> <ul style="list-style-type: none"><li>• CMW: error-string: @ComNbi @SwM::createUpgradePackage UpgradePackage=ERIC-Test-UP 150922123955-P1A01 is still active</li><li>• CMW: error - saImmOmAdminOperationInvoke_2 admin-op RETURNED: SA_AIS_ERR_BUSY (10)</li><li>• CMW: ERROR (cmw-swm): Failed to execute create UP/CSP command for uri: file:///SoftwareManagement/cmw-csp-KQJd-UP2.tgz</li></ul> <p>The result of a successful execution of this action is an action ID. The user can use the action ID together with output of command cmw-swm --up-list to identify the UP ID, which is uniquely associated with each UP/CSP.</p>	<ul style="list-style-type: none"><li>• Create UP from local file path: \$ cmw-swm --up-create /home/UP.tgz</li><li>• Create UP from remote location, protected with password: \$ cmw-swm --up-create sftp://user@hostname/dir/UP_dir pa55w0rd</li></ul>
--up-prepare	Prepare the ME for activation of the UP/CSP.	\$ cmw-swm --up-prepare ERIC-Test-UP150922123955-P1A01
--up-activate	Activate UP/CSP step by step or all at once.	\$ cmw-swm --up-activate ERIC-Test-UP150922123955-P1A01
--up-confirm	Confirm the upgrade procedure on the ME.	\$ cmw-swm --up-confirm ERIC-Test-UP150922123955-P1A01





Table 8 Command Options

Option	Description	Example
--up-cancel	Cancel the upgrade procedure before confirmation or cancel Activate action at breakpoints between steps. This option can also be used to cancel long running actions such as Prepare or Activate.	<b>\$ cmw-swm --up-cancel ERIC-Test-UP150922123955-P1A01</b>
--up-remove	Delete a UP/CSP from the ME.	<b>\$ cmw-swm --up-remove ERIC-Test-UP150922123955-P1A01</b>
--up-status	<p>Get state of specific UP/CSP ID, track the progress and result of actions.</p> <p>Output format:            Action: &lt;action-name&gt; -            &lt;action-state&gt;            Result: &lt;final-result-of-the-action&gt;            State: &lt;UP-state&gt;</p> <p>The last line is only displayed when a UP ID is specified in the command.</p>	<ul style="list-style-type: none"> <li>• Status of the last Create or Remove action. In this case, as UP is not created or deleted, no UP ID can be specified. <b>\$ cmw-swm --up-status</b>            Output:            Action: CreateUpgradePackage - FINISHED            Result: SUCCESS</li> <li>• Status of a UP after other actions:  <b>\$ cmw-swm --up-status ERIC-Test-UP150922123955-P1A01</b>            Output:            Action: Prepare - RUNNING            Result: NOT_AVAILABLE            State: PREPARE_IN_PROGRESS</li> </ul>



Table 8 Command Options

Option	Description	Example
<code>--default-execmethod</code>	<p>View or set the default upgrade execution method.</p> <p>Output format for view: execMethod: &lt;execution-method&gt;</p> <p>The following lines are only displayed when default has been configured for BALANCED execution method: roleId: &lt;ECIM_CrMRole-MO&gt; minSizeOfRole: &lt;minimum-number-of-nodes-in-role&gt; minRemainingCapacity: &lt;minimum-remaining-capacity&gt; numberOfSteps: &lt;Empty&gt;</p> <p>numberOfSteps is not calculated for default.</p>	<ul style="list-style-type: none"> <li>View default execMethod: <code>\$ cmw-swm --default-execmethod</code> Output: execMethod: ROLLING</li> <li>Set default execMethod to ROLLING: <code>\$ cmw-swm --default-execmethod rolling</code></li> <li>Set default execMethod to ONE-STEP: <code>\$ cmw-swm --default-execmethod one-step</code></li> </ul>
<code>--up-execmethod</code>	<p>View or set the upgrade execution method of a UP/CSP.</p> <p>Output format for view: execMethod: &lt;execution-method&gt;</p> <p>The following lines are only displayed when the UP has been configured for BALANCED execution method): roleId: &lt;ECIM_CrMRole-MO&gt; minSizeOfRole: &lt;minimum-number-of-nodes-in-role&gt; minRemainingCapacity: &lt;minimum-remaining-capacity&gt; numberOfSteps: &lt;calculated-number-of-steps&gt;</p>	<ul style="list-style-type: none"> <li>View UP execMethod: <code>\$ cmw-swm --up-execmethod ERIC-Test-UP150922123955-P1A01</code> Output: execMethod: BALANCED roleId: roleId=ScalableRole,ECIM_CrMcrMid=1 minSizeOfRole: 2 minRemainingCapacity: 35 numberOfSteps: 5</li> <li>Set UP execMethod to ROLLING: <code>\$ cmw-swm --up-execmethod ERIC-Test-UP150922123955-P1A01 rolling</code></li> <li>Set UP execMethod to BALANCED using Role: <code>\$ cmw-swm --up-execmethod ERIC-Test-UP150922123955-P1A01 roleId=ScalableRole,ECIM_CrMcrMid=1 8 45</code> (Minimum 8 nodes and minimum 45% remaining capacity.)</li> </ul>

Example of upgrading software using cmw-swm:

1. Create a UP/CSP from the file argument:



```
cmw-swm --up-create /storage/system/software/coremw/repository/
UP/ERIC-ActivateCommand_UP/ERIC-ActivateCommand_UP.tgz
```

The UP/CSP can be created from file://, sftp://, or path of file.

2. Verify that the UP/CSP was created and to get the <UP-ID> of the UP/CSP:

```
cmw-cwm --up-list
```

3. View the progress and result of the command in the previous step:

```
cmw-swm --up-status
```

4. View the UP status:

```
cmw-swm --up-status <UP-ID>
```

5. Verify that the UP has state INITIALIZED.

6. Set the UP/CSP upgrade execution method to ONE-STEP, ROLLING, or BALANCED:

```
cmw-swm --up-execmethod <UP_ID> one-step | rolling |
<role-name> <min-size-of-role> <min-remaining-capacity>
```

Command `cmw-swm --up-execmethod <UP-ID>` (without execution method arguments) can also be used to view the upgrade execution method configured for the UP/CSP.

7. Prepare the ME for the activation of the UP/CSP with <UP-ID>:

```
cmw-swm --up-prepare <UP-ID>
```

The prerequisite of this action is that the state of the UP/CSP is INITIALIZED. This step is considered completed when the state of the UP/CSP is PREPARE\_COMPLETED.

8. Activate the UP/CSP step by step or all at once:

```
cmw-swm --up-activate <UP-ID>
```

If the UP/CSP is designed with more than one step and `ignoreBreakpoints` is set to `false`, action `Activate` is called several times.

9. Verify that the upgrade procedure is completed:

```
cmw-swm --up-confirm <UP-ID>
```

The expected state is COMMIT\_COMPLETED.

10. Delete the UP/CSP MO given as action parameter:

```
cmw-swm --up-remove <UP-ID>
```



This command also deletes all files temporarily stored in the ME and associated with the UP/CSP.

During consuming of the CSP 1.2 or above where the campaign is generated automatically, multiple campaigns are sometimes generated. If this occurs, the preparation of the CSP fails with the following status:

```
Action: Prepare – FINISHED  
Result: FAILURE - Unsupport multiple tcg campaigns yet  
State: INITIALIZED
```

ECIM SwM does not support multiple campaigns, so the `cmw-ait-install` command is provided for the Automatic Installation Tool (AIT) to continue consuming this package at installation time:

```
cmw-ait-install [--status] | [--disable-backup]] <CSP-package>
```

```
cmw-ait-install [--status] | [--disable-backup]] <CSP-ID>
```

**Note:** This command is only to be executed to the CSP that has just failed the prepare phase with the above status.



Table 9 Command Options

Option	Description	Example
<code>--disable-backup</code>	By default, a backup is automatically created before consuming the package. This option is to disable the backup creation.	Check the CSP status: \$ <b>cmw-swm --up-status cba.test-CXP123456-1.0.0</b> Output: Action: Prepare – FINISHED Result: FAILURE - Unsupport multiple tcg campaigns yet State: INITIALIZED  Continue the CSP: \$ <b>cmw-ait-install --disable-backup cba.test-CXP123456-1.0.0</b>
<code>--status</code>	Get state of specific CSP ID, track the progress and result of actions.  Output format: State: <CSP-state> Action: <action-name> Additional Info: [Additional information of the current state, if applicable] Result: <action-result> Result Info: [Informa tion about result, if applicable]	Check the AIT installation status: \$ <b>cmw-ait-install --status cba.test-CXP123456-1.0.0</b> Output: State: RUNNING Action: PREPARING Additional Info: Generating campaigns from CSM model Result: NOT_AVAILABLE Result Info:

## 5.7 Configure Reboot Behavior of Single-Step Procedures

SMF single-step procedures are typically used for installation (or removal) of components/applications where a component/application is not in-service upgradeable. If a single-step upgrade procedure contains software bundles, which require a reboot to install or remove, SMF can either perform a cluster reboot or perform nodewise reboot of only the affected nodes.

The default SMF behavior is to perform nodewise reboot during single-step upgrade procedures.

To define the affected nodes for nodewise reboot, the `campaign.xml` file must specify them in the `<singleStepUpgrade>` section, as shown in Example 68.



```
<singleStepUpgrade>
  <upgradeScope>
    <forAddRemove>
      <deactivationUnit>
        <actedOn>
          <byName objectDN="safAmfNode=SC-2,safAmfCluster=myAmfCluster"/>
        </actedOn>
      </deactivationUnit>
      <activationUnit>
        <actedOn>
          <byName objectDN="safAmfNode=SC-2,safAmfCluster=myAmfCluster"/>
        </actedOn>
      </activationUnit>
    </forAddRemove>
  </upgradeScope>
</singleStepUpgrade>
```

Example 68 Campaign Defining Affected Nodes for Single-Step Upgrade Procedures

Attribute `smfSSAaffectedNodesEnable` of the SMF configuration controls the nodewise reboot feature and defaults to 1. To change SMF behavior to reboot all nodes in the cluster during a single-step upgrade procedure, a campaign must disable nodewise reboot, as shown in Example 69.

```
<campWrapupAction>
  <immCCB ccbFlags="0">
    <modify operation="SA_IMM_ATTR_VALUES_REPLACE" objectDN="smfConfig=1,safApp=safSmfService">
      <attribute name="smfSSAaffectedNodesEnable" type="SA_IMM_ATTR_SAUINT32T">
        <value>0</value>
      </attribute>
    </modify>
  </immCCB>
</campWrapupAction>
```

Example 69 Campaign Disabling Nodewise Reboot Feature



## 6 Configuring ISP and Remote Fencing

### 6.1 Configure ISP Events for Cluster Restarts

ISP events for cluster restarts are disabled by default.

This feature can be enabled or disabled during AIT installation of Core MW. For more information, see Section [Automatically Installing Core MW by Using AIT in Core MW SW Installation](#).

ISP events for cluster restarts can be enabled in runtime with the `cmw-utility` command as follows:

```
cmw-utility immcfg --attribute ispClusterRebootLogEnabled=1  
CmwMonitorIspId=1,CmwMonitorId=1,CmwSysConfigId=1
```

ISP events for cluster restarts can be enabled within a campaign by setting attribute `ispClusterRebootLogEnabled` of object with class type `CmwMonitorIsp` to 1.

An example of a campaign Init section that enables ISP events for cluster restarts by setting attribute `ispClusterRebootLogEnabled` to 1 is shown in Example 70.

```
<campInitAction>  
<immCCB ccbFlags="0">  
  <modify objectDN="CMW_GETDN(^CmwMonitorIspId=1,CmwMonitorId=1,=>  
CmwSysConfigId=1$)" operation="SA_IMM_ATTR_VALUES_MODIFY">  
    <attribute name="ispClusterRebootLogEnabled" type="SA_IMM_ATTR_SAUINT32T">  
      <value>1</value>  
    </attribute>  
  </modify>  
</immCCB>  
</campInitAction>
```

Example 70 ISP Events for Cluster Restarts

**Note:** In order for cluster restart events to be visible in the ISP report, the Core MW `ISP_REPORT` feature must also be enabled, as described in Section [3.10.4 In-Service Performance Report Feature](#) on page 19.

### 6.2 Configure Remote Fencing Parameters

All RF parameters are configured within the RF plug-in. For more information on RF plug-ins, refer to Section [Remote Fencing in Core MW Programmer's Guide](#).

#### 6.2.1 OpenSAF Parameters

Table 10 describes the OpenSAF fencing parameters.



Table 10 OpenSAF Fencing Parameters

Parameter	Description	Example
FMS_NODE_ISOLATION_TIMEOUT	Enable self-fencing set value to non-zero value. Unit in milliseconds.	Export FMS_NODE_ISOLATION_TIMEOUT=10
FMS_PROMOTE_ACTIVE_TIMER	The Promote active timer is set to delay the Standby controllers reboot request, as the Active controller also requests reboot of the standby. The resolution is in 10 ms units.	Export FMS_PROMOTE_ACTIVE_TIMER=300
FMS_USE_REMOTE_FENCING	Enable remote fencing.	Export FMS_USE_REMOTE_FENCING=1
FMS_FENCE_CMD	Fencing command.	Export FMS_FENCE_CMD="stonith"
FMS_DEVICE_TYPE	Fencing device type.	Export FMS_DEVICE_TYPE="external/libvirt"
FMS_HYPERVISOR_URI	Connection URI to hypervisor host machine.	Export FMS_HYPERVISOR_URI="qemu+tcp://192.168.122.1/system"
FMS_FENCE_ACTION	Fence action.	Export FMS_FENCE_ACTION="reset"

## 6.2.2 Core MW Parameters

Table 11 describes the Core MW RF parameters.

Table 11 Core MW RF Parameters

Parameter	Description	Example
FMS_USE_REMOTE_FENCING	Enable remote fencing.	Export FMS_USE_REMOTE_FENCING=1
RF_NAME	Remote fencing plug-in name.	Export RF_NAME="FENCING_STONITH-CXP9010009_1"
RF_VERSION	Remote fencing plug-in version.	Export RF_VERSION="R1A01"
TARGET_PLUGIN	Full path location to target.<plugin> file.	Export TARGET_PLUGIN="/cluster/home/fencingtest/target.FENCING_STONITH-CXP9010009_1-R1A01"





## 7 Performance Management

PM jobs can be created, modified, deleted, started, and stopped programmatically using the IMM Object Management (OM) API.

This section describes the following shell command support for managing ECIM PM Jobs:

- Creating an ECIM PM Job
- Starting an ECIM PM Job
- Stopping an ECIM PM Job
- Deleting an ECIM PM Job
- Reporting status of an ECIM PM Job
- Listing ECIM PM Jobs
- Modifying an ECIM PM Job

This section also describes the following PM operation:

- Displaying active PM values for an instance

Commands are to be run as root user or prefixed with `sudo` and run by a user belonging to the `system-adm` group.

**Note:** These procedures require the Performance Management module in Cloud Enabled systems.

### 7.1 Create ECIM PM Job

To create an ECIM PM Job, use the `cmw-pmjob-create` command.

The command syntax is as follows:

```
cmw-pmjob-create <job-name>
    (-u <pmgroup-id>
    [-M <meas-reader>]
    [-R <variation-rate>]
    [-d <threshold-direction>]
    [(-I <mo-instance>...)]
    [-m <meas-type>]
    [-T <threshold-id>
    -H <threshold-high>
    -L <threshold-low>
    -S <severity>]
    )...
```



```

[-t <job-type>]
[-r <reporting-period>]
[-g <granularity-period>]
[-p <job-priority>]
[-q <requested-state>]
[-c <job-control>]

```

The syntax is interpreted as follows:

- job-name is attribute pmJobId in the PmJob MO.
- pmgroup-id is the MO class (moClass) name of the Measurement Type (MT).
- meas-type is the MT.
- The part within ( ) can occur multiple times, as indicated with three dots.

The command supports any number of Measurement Readers (MRs) and Measurement Specifications (MSs).

For threshold jobs, up to four thresholds object are supported.

The MS can use a direct reference to an MT or use an indirect reference by referring to a moClass.

A direct reference to an MT occurs when the moClass (option -u) and the MT (option -m) are specified for the MR.

An indirect reference to a group occurs when only the moClass (option -u) is specified for the MR. However, indirect references are only valid for measurement jobs (-t 1), not for threshold jobs (-t 2).

The optional parameters for threshold jobs are as follows:

- -T <threshold-id>
    - -H <threshold-high>
    - -L <threshold-low>
    - -S <severity>
      - 3 – CRITICAL
      - 4 – MAJOR
      - 5 – MINOR (default value)
      - 6 – WARNING
  - -R <variation-rate>
    - 0 – PER\_SECOND (default value)
    - -1 – PER\_GP
- Note:** Only used for Cumulative Counter (CC) collection method.
- -d <threshold-direction>
    - 1 – INCREASING



- 2 – DECREASING

The optional Job parameters are as follows:

— -i <mo-instance>

A specific MT instance to be referenced. If not specified, all MT instances are referenced. It supports any number of instance parameters. The format is as follows: -i <instance1> -i <instance2> -i <instance3>.

— -t <job-type>

- 1 – MEASUREMENTJOB (default value)
- 2 – THRESHOLDJOB

— -r <reporting-period>

- 3 – 1 minute
- 4 – 5 minutes
- 5 – 15 minutes (default value)
- 6 – 30 minutes
- 7 – 1 hour
- 8 – 12 hours
- 9 – 24 hours

— -g <granularity-period>

- 3 – 1 minute
- 4 – 5 minutes
- 5 – 15 minutes (default value)
- 6 – 30 minutes
- 7 – 1 hour
- 8 – 12 hours
- 9 – 24 hours

— -p <job-priority>

- 1 – LOW
- 2 – MEDIUM (default value)
- 3 – HIGH

— -q <requested-state>

- 1 – ACTIVE (default value)
- 2 – STOPPED

— -c <job-control>

- 0 – FULL (default value)
- 1 – STARTSTOP
- 2 – VIEWONLY

— -G <job-group>

— -x <compression-type>

- 0 – GZIP

The creation of an ECIM PM Job named job3, which directly references MT rx0ctets in moClass mocRx is shown in Example 71.

10. (1)  $\frac{1}{2}$  (2)  $\frac{1}{2}$  (3)  $\frac{1}{2}$  (4)  $\frac{1}{2}$  (5)  $\frac{1}{2}$  (6)  $\frac{1}{2}$  (7)  $\frac{1}{2}$  (8)  $\frac{1}{2}$  (9)  $\frac{1}{2}$  (10)  $\frac{1}{2}$

### Example 71 Creating an ECIM PM Job

The command returns exit code 0 on success, otherwise exit code 1.

Attempting to create an ECIM PM Job without first defining the MT or moClass results in failure, as shown in Example 72.

[illegible]

### Example 72 Failed Attempt to Create an ECIM PM Job

How to create an ECIM PM Job that references an MT indirectly by the moClass `moRxRate` (by not specifying option `-m`) is shown in Example 73.

10.  $f(x) = 4$  dan  $g(x) = 3x + 2$  pada  $x = 2$  menyatakan

### Example 73 Creating an ECIM PM Job That Indirectly References an MT

Multiple MT references can be applied to the same job, as shown in Example 74.

[illegible]

### Example 74 Creating an ECIM PM Job with Multiple MTs

job5 is defined as a threshold type job (flag -t is set to 2), as shown in Example 74.



job5 contains two MR references that can be described as follows:

- The first MR directly references an MT named `rxDiscFr` in moClass `moRx`. It defines a `PmThresholdMonitoring` threshold named `rxDiscAlm` of severity class 3 (CRITICAL). The high threshold is set to 300 and the low threshold is set to 250.
- The second MR directly references an MT named `rxRate` in the moClass `moRxRate`. It defines three `PmThresholdMonitoring` thresholds (`rxRateOversub`, `rxRateHigh`, and `rxRateBusy`), each with an assigned severity and threshold levels.

Attribute `job-control` is optional. Defaults to FULL, which means that the job can be started, stopped, or deleted.

PMJob2 in Example 75 is defined as a measurement job, which has `job-control` as 1 (STARTJOB).



Example 75 Creating an ECIM PM Job with Start Stop Control

## 7.2 Start ECIM PM Job

To start an ECIM PM Job, use the `cmw-pmjob-start` command.

**Note:** The precondition is that `job-control` is FULL or STARTSTOP.

The command syntax is as follows:

**cmw-pmjob-start** <job-name>

The syntax is interpreted as follows:

- `job-name` is attribute `pmJobId` in the PmJob MO.



Example 76 Starting an ECIM PM Job

The command returns exit code 0 on success, otherwise exit code 1.

A failure to start an ECIM PM Job whose `job-name` does not exist is shown in Example 77.



```
CMW-PMJOB-START: 2018-06-01 10:00:00.000
CMW-PMJOB-START: 2018-06-01 10:00:00.000
CMW-PMJOB-START: 2018-06-01 10:00:00.000
CMW-PMJOB-START: 2018-06-01 10:00:00.000
```

#### Example 77 Starting a Non-Existing ECIM PM Job

Attempting to start an already started ECIM PM Job results in exit code 0 with the following message:

```
Warning: PM Job <job-name> is already Active
```

## 7.3 Stop ECIM PM Job

To stop an ECIM PM Job, use the **cmw-pmjob-stop** command.

**Note:** The precondition is that `job-control` is FULL or STARTSTOP.

The command syntax is as follows:

```
cmw-pmjob-stop <job-name>
```

The syntax is interpreted as follows:

— `job-name` is attribute `pmJobId` in the `PmJob` MO.

```
CMW-PMJOB-STOP: 2018-06-01 10:00:00.000
CMW-PMJOB-STOP: 2018-06-01 10:00:00.000
CMW-PMJOB-STOP: 2018-06-01 10:00:00.000
```

#### Example 78 Stopping an ECIM PM Job

The command returns exit code 0 on success, otherwise exit code 1.

Attempting to stop an already stopped ECIM PM Job results in exit code 0 with the following message:

```
Warning: PM Job <job-name> is already Stopped
```

## 7.4 Delete ECIM PM Job

To delete a stopped ECIM PM Job, use the **cmw-pmjob-delete** command.

**Note:** The precondition is that `job-control` is FULL.

The command syntax is as follows:

```
cmw-pmjob-delete <job-name>
```



The syntax is interpreted as follows:

- `job-name` is attribute `pmJobId` in the `PmJob` MO.

Example 79

#### Deleting an ECIM PM Job

The command returns exit code 0 on success, otherwise exit code 1.

Attempting to delete an ACTIVE ECIM PM Job results in exit code 1 with the following message:

```
CMW: ERROR (cmw-pmjob-delete): PM Job <job-name> is
Active - can not be deleted
```

Attempting to delete a non-existent ECIM PM Job results in exit code 1 with the following message:

```
CMW: ERROR (cmw-pmjob-delete): PM Job <job-name> does not exist
```

## 7.5

### Report Status of ECIM PM Job

To report the status of an ECIM PM Job, use the `cmw-pmjob-status [-v] <job-name>` command.

The command syntax is as follows:

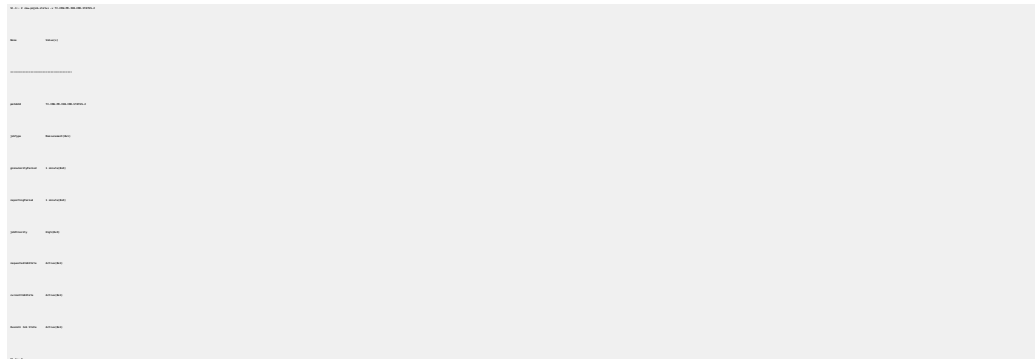
```
cmw-pmjob-status [-v] <job-name>
```

The syntax is interpreted as follows:

- `job-name` is attribute `pmJobId` in the `PmJob` MO.
- The optional `-v` flag produces a verbose output. Omitting the `-v` flag only produces the job name and status.

Example 80

#### Reporting Status of an ECIM PM Job



Job Name	Requested Job State	Current Job State
Job 1	Running	Running
Job 2	Waiting	Waiting
Job 3	Completed	Completed
Job 4	Failed	Failed
Job 5	Cancelled	Cancelled
Job 6	On Hold	On Hold
Job 7	Paused	Paused
Job 8	Restarting	Restarting
Job 9	Terminated	Terminated
Job 10	Unknown	Unknown

#### Example 81 Reporting Status of an ECIM PM Job Using Verbose Mode

If the job does not exist or if the status cannot be shown, the command returns exit code 1 and logs the following message:

```
CMW: ERROR (cmw-pmjob-status): Could not show the status  
for PM Job <job-name>
```

Otherwise the command returns exit code 0.

## 7.6 List ECIM PM Jobs

To list ECIM PM Jobs, use the **cmw-pmjob-list** command.

The command syntax is as follows:

```
cmw-pmjob-list [<options>]
```

The possible options are as follows (<N> is a numeral):

- **-v** – Use long format for each job.
- **-h** | **--help** – Display this help and exit.
- **-f** – List only jobs with problems (faulty).
- **-r** <N> – List only jobs with this reporting period.
- **-t** <N> – List only jobs with this job type.
- **-p** <N> – List only jobs with this job priority.
- **-g** <N> – List only jobs with this granularity period.
- **-q** <N> – List only jobs with this requested job state.
- **-s** <N> – List only jobs with this current job state.

Using the command without any options results in a short listing, showing each ECIM PM Job name, requested job state, and current job state, as shown in Example 82.





```
SC-1:~ # cmw-pmjob-list
pmJobId=TCPMJCA_0, requestedJobState=Stopped(0x2), currentJobState=Stopped(0x2)
pmJobId=TCPMJCA_1, requestedJobState=Active(0x1), currentJobState=Active(0x1)
pmJobId=TCPMJCA_2, requestedJobState=Stopped(0x2), currentJobState=Stopped(0x2)
pmJobId=TCPMJCA_3, requestedJobState=Active(0x1), currentJobState=Active(0x1)
pmJobId=TCPMJCA_4, requestedJobState=Stopped(0x2), currentJobState=Stopped(0x2)
SC-1:~ #
```

### Example 82 Listing ECIM PM Jobs

The command contains a `-v` flag for verbose (long format). The output contains detailed information about each ECIM PM Job, as shown in Example 83.

### Example 83 Listing ECIM PM Jobs Using Verbose Mode

Multiple options can be combined in one command to **OR** the result. For example, using multiple options on the same job list as in Example 83, jobs with `reportingPeriod` equal to 1 minute and `jobType` equal to Measurement can be shown. A filtered list using multiple options is shown in Example 84.



Example 84 Listing ECIM PM Jobs Using Multiple Options

Multiple options can be used in short format (without flag -v), as shown in Example 85.

Example 85 Listing ECIM PM Jobs in Short Format with Multiple Flags

## 7.7 Modify ECIM PM Job

To modify a stopped ECIM PM Job, use the `cmw-pmjob-modify` command.

The command syntax is as follows:

```
cmw-pmjob-modify <job-name> <added-M0s> | <deleted-M0s> |  
<modified-attributes>
```



Syntax when adding MeasurementReader MOs:

```
cmw-pmjob-modify <job-name>
  [(-C
    [-M <meas-reader>]
    [-R <threshold-rate-of-variation>]
    [-d <threshold-direction>]
    [(-i <mo-instance>...)]
    -u <pmgroup-id>
    [-m <meas-type>]
  )...]
```

Syntax when adding PmThresholdMonitoring MOs:

```
cmw-pmjob-modify <job-name>
  [(-C
    [-M <meas-reader>]
    (-T <threshold-id>
      -H <threshold-high>
      -L <threshold-low>
      -S <severity>)
  )...]
```

Syntax when adding both MeasurementReader and PmThresholdMonitoring MOs:

```
cmw-pmjob-modify <job-name>
  [(-C
    [-M <meas-reader>]
    [-R <threshold-rate-of-variation>]
    [-d <threshold-direction>]
    [(-i <mo-instance>...)]
    -u <pmgroup-id>
    -m <meas-type>
    (-T <threshold-id>
      -H <threshold-high>
      -L <threshold-low>
      -S <severity>)
  )...]
```

Syntax when deleting MOs:

```
cmw-pmjob-modify <job-name>
  [(-D
    -M <meas-reader>
    [-T <threshold-id>]
  )...]
```

Syntax when modifying attributes:

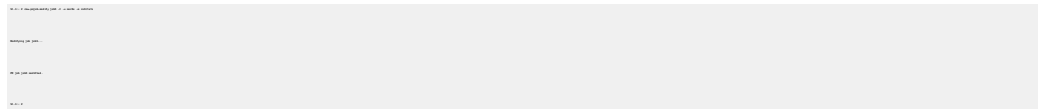
```
cmw-pmjob-modify <job-name>
  [-r <reporting-period>]
```



```
[-g <granularity-period>]
[-p <job-priority>]
[-G <job-group>]
[-x <compression-type>]
```

This command requires -C to indicate child MO creation, and -D to indicate child MO deletion. All the other options have the same meanings as with the `cmw-pmjob-create` command, see Section 7.1 Create ECIM PM Job on page 53. The parts within () can occur multiple times.

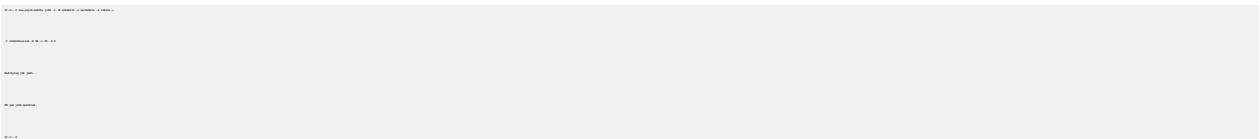
The addition of a `MeasurementReader` MO to an existing job named `job3` is shown in Example 86.



#### Example 86 Adding a `MeasurementReader` MO to an Existing ECIM PM Job

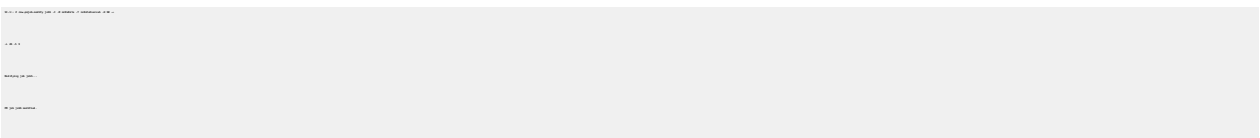
The command returns exit code 0 on success, otherwise exit code 1.

The addition of both a `MeasurementReader` MO and its child `PmThresholdMonitoring` MO to an existing threshold job named `job5` is shown in Example 87.



#### Example 87 Adding a `MeasurementReader` MO and its Child `PmThresholdMonitoring` MO to an Existing ECIM PM Job

The addition of a `PmThresholdMonitoring` MO to an existing `MeasurementReader` MO of an existing job named `job5` is shown in Example 88. As the `PmThresholdMonitoring` MO is a child of a `MeasurementReader` MO, parameter -M must be specified.



#### Example 88 Adding a `PmThresholdMonitoring` MO to an Existing `MeasurementReader` MO

The deletion of a `MeasurementReader` MO from an existing job named `job3` is shown in Example 89.



```

$ cmw-pm-show-counters -j job5 -M MeasurementReader

```

#### Example 89 Deleting a MeasurementReader MO from an Existing ECIM PM Job

The deletion of a PmThresholdMonitoring MO from an existing threshold job named job5 is shown in Example 90. Parameter -M must be specified.

```

$ cmw-pm-show-counters -j job5 -M PmThresholdMonitoring

```

#### Example 90 Deleting a PmThresholdMonitoring MO from an Existing ECIM PM Job

Modifying one or more ECIM PM Job attributes for an existing job named job3 is shown in Example 91.

```

$ cmw-pm-show-counters -j job3 -M PmThresholdMonitoring

```

#### Example 91 Modifying Attributes for an Existing ECIM PM Job

## 7.8 Display Active PM Instance Values

The `cmw-pm-show-counters` command can be used to display PM counter values. The command will behave differently based on the CoreMW configuration of `SHOW_COUNTERS_VERSION`.

Version 1 is only applicable to Core MW Classic deployments.

Version 2 is the default and only option we have in Core MW Cloud Enabled deployments.

When version 1 is selected (available only in Core MW Classic deployments), the command displays the current values of all MTs associated with the specified instance.

**Note:** The precondition is that the instance is associated with the MTs that are measured by an active ECIM PM Job.

The command syntax is as follows:

```

cmw-pm-show-counters [-v] <instance>
                    [-t <timeout>]
                    [-j <job-name> [<job-name>...]]
                    [-m <meas-type> [<meas-type>...]]

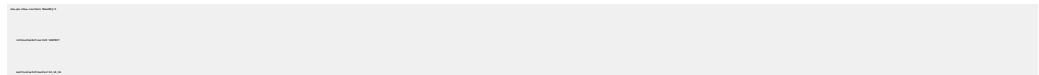
```



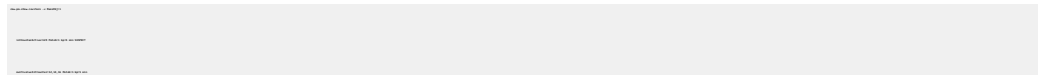
The syntax is interpreted as follows:

- `-v` | `--verbose` is verbose.
- `-t` | `--timeout` `<seconds>` is the time-out. Defaults to 6 seconds.
- `job-name` is attribute `pmJobId` of an active `PmJob` MO. If job names are specified, only instance values from the specified jobs are displayed.
- `meas-type` is attribute `measurementTypeId` of a `MeasurementType` MO. If MTs are specified, only instance values associated with the specified MTs are displayed.

The command returns exit code 0 on success, otherwise exit code 1.



#### Example 92 Displaying Current Instance Values



#### Example 93 Displaying Verbose Current Instance Values

When version 2 is selected (default for Core MW Cloud Enabled deployments), the command will display real time counter instances based on the constraints provided in the command. In this case the instances do not need to be associated with an active job.

The constraints are as follows:

```
cmw-pm-show-counters [<instance>]
                    [-t <timeout>]
                    [-u <pm_group_id> [<pm_group_id> ... ] ]
                    [-m <meas_type> [<meas_type> ...] ]
                    [-f (INSTANCE-MT | MT-INSTANCE | GROUP-INSTANCE-MT | GROUP-MT-INSTANCE
```

- `-t` | `--timeout` `<seconds>` is the time-out. Default value is 6.
- `pm_group_id` and `meas_type_id` may be specified explicitly or as a wildcard expression.
- Format options allow the user to group the output results according to their requirements. The default format is `INSTANCE-MT`.

**Note:** The wildcard filter construct is `*` and matches zero or more characters including null string. The wildcard must also be escaped.



```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

```
SELECT * FROM SYS.DATABASES WHERE NAME = 'tempdb'
```

**Example 94**    Displaying real time instance values using specific instance name and instance name with wildcard.







## 8 OpenSAF Tools

### 8.1 OpenSAF Tools Wrapper

The `cmw-utility` command can be used to run the following OpenSAF tools:

- `immfind`
- `immlist`
- `immcfg`
- `immadm`
- `amfadm`
- `runasroot`

The command syntax is as follows:

```
cmw-utility [-h | --help] <immfind | immlist | immcfg | immadm |  
amfadm | runasroot> [<options>]
```

The syntax is interpreted as follows:

- `-h | --help` – Print help menu.
- `options` – See the following sections for the individual OpenSAF tools.

#### 8.1.1 `immfind`

`immfind` is used for searching for IMM objects.

The command syntax is as follows:

```
cmw-utility immfind [path...] [<options>]
```

The options are as follows:

- `-c | --class <name>` – Only search for objects of the specified class.
- `-t | --timeout <timeout>` – Utility time-out in seconds.

---

Example 95 Searching for IMM Objects



### 8.1.2 immlist

`immlist` is used for listing IMM objects.

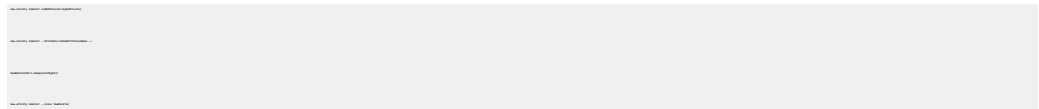
The command syntax is as follows:

```
cmw-utility immlist [<options>] <object-name> [<object-name>...]
```

```
cmw-utility immlist [<options>] <class-name>
```

The options are as follows:

- `-a` | `--attribute <name>` – Attribute name to print.
- `-c` | `--class <name>` – Class definition to print.
- `-t` | `--timeout <timeout>` – Utility time-out in seconds.



Example 96 Listing IMM Objects

### 8.1.3 immcfg

`immcfg` is for creating, deleting, or modifying IMM objects.

The command syntax is as follows:

```
cmw-utility immcfg ([<options>] [<object-name>])...
```

The options are as follows:

- `-a` | `--attribute <name>[+|-]=<value>` – Change attribute.
- `-c` | `--create-object <class-name>` – Create an object of the specified class.
- `-d` | `--delete-object [<object-name>]...` – Delete objects.
- `-m` | `--modify-object [<object name>]...` – Modify objects.
- `-u` | `--unsafe` – The CCBs generated by `immcfg` have `SA_IMM_CCB_REGISTERED_OI` set to `false`, allowing CCB commit when Object Implementers (OIs) are missing.
- `-t` | `--timeout <timeout>` – Utility time-out in seconds.



#### Example 97 Creating IMM Objects

#### Example 98 Changing IMM Objects

#### Example 99 Deleting IMM Objects

### 8.1.4

#### **immadm**

**immadm** is used for performing an IMM admin operation.

The command syntax is as follows:

**cmw-utility immadm [<options>] [<object-name>...]**

The options are as follows:

- **--disable-tryagain** – Disable try again handling. Defaults to no.
- **-o | --operation-id <id>** – Numerical operation ID.
- **-p | --parameter=<p>** – Parameter/parameters to admin operation.

Parameter syntax: **<name>:<type>:<value>**

Value types according to **imm.xsd**: **SA\_INT32\_T, SA\_UINT32\_T, SA\_INT64\_T, SA\_UINT64\_T, SA\_TIME\_T, SA\_NAME\_T, SA\_FLOAT\_T, SA\_DOUBLE\_T, SA\_STRING\_T.**

- **-t | --timeout <timeout>** – Utility time-out in seconds.

#### Example 100 Performing an IMM Admin Operation

### 8.1.5

#### **amfadm**

**amfadm** is used for performing an AMF admin operation.

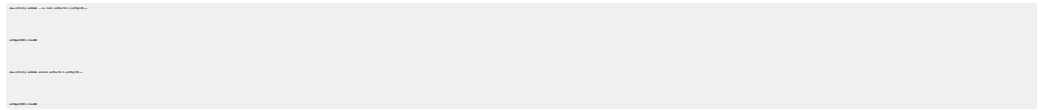
The command syntax is as follows:

**cmw-utility amfadm [<options>] <command> <object-name>**



The options are as follows:

- `-s` | `--su` – Specified object DN is checked to be an SU before performing AMF admin operation.
- `-t` | `--timeout` `<timeout>` – Utility time-out in seconds.
- `<command>` – `lock`, `unlock`, `lock-in`, `unlock-in`, `shutdown`, `restart`, `si-swap`, `sg-adjust`, `repaired`, `eam-start`, `eam-stop`.



Example 101 Performing ADM Admin Operations

## 8.1.6

### **runasroot**

`runasroot` is used for running a command with root privileges. It only supports the SMF service and must be called through SMF callbacks.

The command syntax is as follows:

**`cmw-utility runasroot <command>`**

For an example, refer to Section Built-in SMF API Callback Receiver in *Core MW Software Management*.



## 9 Backup and Restore Framework (Core MW Classic)

For information about the Backup and Restore Framework, refer to BRF-C Management Guide.